The Development of a Genetic Algorithm for Real Time Water Allocation and Water Scheduling in Complex Irrigation Systems

Kampanad Bhaktikul

A thesis submitted for the degree of Doctor of Philosophy

School of Civil and Environmental Engineering

The University of Edinburgh

March 2001

ABSTRACT

The demand for water is increasing rapidly in the developing world. The gap between water demand and water supply is widening, making effective water resources management vital. Irrigation represents the largest use of fresh water globally, and improved management of water within irrigation systems is essential if scarce resources are to be used in an equitable manner and to their maximum benefit. The management of water allocation in large irrigation systems is important, as even a small improvement in operation may lead to significant benefits.

An optimisation approach based on Genetic Algorithms (GAs) is developed for real time allocation of irrigation water supplies. Appropriate objective functions for the water allocation problem have been derived previously and solved using quadratic programming (QP). There had been concerned that the QP approach may become computationally bounded for large systems. It was also thought that the approach would be difficult to apply to water scheduling problems. This research describes work on the development of a GA for the water allocation problem. Although GAs have been actively researched for 30 years, only one previous application to an irrigation problem has been found in the literature. The GA approach is very flexible, and is easily set up for a wide range of linear and non-linear objective functions. In developing the GA solver the intention was to have a generic code easily adapted and used. This has been achieved and the same core routines are used for a wide range of problems.

Applied to the water allocation problem, the GA approach can provide solutions that are similar to those produced by QP. It is, however, sensitive to string length, and has difficulty in meeting nodal water balance constraints. It is concluded that the GA approach offers no advantage over QP for the water allocation problem.

Further development of a Genetic Algorithm (GA) to solve an irrigation water scheduling problem is described. The objective is to optimise the utilisation of water resources during water stress periods in irrigation systems operating on a rotational basis. Objective functions for the water scheduling problem are developed. Solutions are presented using a GA, and the

advantages and shortcomings of different approaches are discussed. The objective is to minimise the irrigation water supply to the system when adequate supplies exist, and to distribute crop stress in an equitable manner in periods of water shortage. It was demonstrated that a formulation called the "Zero-1" approach was most effective in solving the problem, performing significantly better than traditional systems, and more recent scheduling developments. A number of practical applications are presented that demonstrate the effectiveness of the GA approach.

The GA approach is relatively simple to set up and is very robust in application to water scheduling problems. It could offer significant benefits to irrigation system planners and managers.

ACKNOWLEDGEMENTS

My study experience in UK, especially in Edinburgh, was a rewarding one. I sincerely thank those persons and institutions who tangibly and intangibly contributed to my study and to my comfortable stay, as follows:

First and foremost I am indebted to Dr. Robin Wardlaw for his excellent supervision, patient advise and guidance.

I would like to thank the National Energy Policy Office of Thailand who awarded me the Royal Thai Government Scholarship, by cooperation with Civil Service Commission, Bangkok, and Thai Government Students'Office, London. This support was essential for my study and to the development of my research.

I also thank all staffs of the School of Civil and Environmental Engineering for their help, and especially Miss Dawn Potter for administrative assistance and Mr. Chris Burnside for his computing expertise.

Many thanks are to Sharif and May for their valuable guidance on software development.

Thanks are due to School of Irrigation, Royal Irrigation Department, Thailand, for my back ground in Irrigation Engineering, and to Faculty of Environment and Resource Studies, Mahidol University in Technology of Environmental Management.

Sincere thanks also to the good and helpful friends I have met, such as, Pete, Kevin, Richard, Jie, Mark, Ton, Poom, Aun, Nop, and other friends from Thai Society of Edinburgh, friends from Theravada Group, friends at 64 West Mains, and all my students who always keep in touch.

Finally, I would like to take this opportunity to thank my parents, especially "Mum", who gave life, love and understanding at any time, and to my brother for encouragement with spirit for their support and help throughout the years of the study.

CONTENTS

PUBLICATIONS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

CCA culturable command area.

FSL full supply level.

1. INTRODUCTION

1.1 General

The utilisation of natural resources must be sustainable, and increasingly ways are being sought to bring about effective management of diminishing resources. Natural resources must be used economically and sustainably, if benefits are to be maximised and negative feedback to the ecological system minimised. Rigorous planning and management are essential for long term sustainable development.

Water was once considered "free goods" in economic terms. However, with increasing population pressures and development activities, the available per-capita resource is diminishing both in terms of quantity and quality. The catchment is the basic unit for water resource assessment. At this scale environmental change affects management of the water resource, and impacts on the people dependent on that resource. Various problems such as, deforestation, agricultural extension, population growth, urbanisation and industrial expansion directly effect water availability and water quality. Under these pressures, there is increasing need for careful water and environmental management.

Irrigation represents the largest use of fresh water globally. It is an ancient art. Irrigation is thought to have begun over 5000 years ago. Hieroglyphics showing the opening of an irrigation canal dates from 3500 BC (Israelsen and Hansen 1962). There is evidence that many ancient civilisations had meaningful irrigation systems. Evidence exists of irrigation in Mesopotamia from about 3000 BC, in China from 2000 BC, and in India from 300 BC. Systems in Sri Lanka and Bali are over 1000 years old (Spiertz 1991), and there are systems in northern Thailand that are over 700 years old. When these systems were developed, population pressures were very much lower than they are today.

The world irrigated area expanded from 94 million ha to about 220 million ha between 1950 and 1984. The annual expansion rate was 4.1% from 1950 to 1960, 3.5% from 1960 to 1970, and 2.3% from 1974 to 1984. Expansion in irrigated area since 1960 has been in the developing world (Jensen et al. 1990).

Exponential population growth has effected a similar exponential growth in food demand and indeed in production (Jensen et al. 1990). Thus, the demand for water is increasing rapidly in the developing world due to rapid population growth, industrial growth and agricultural extension. The gap between water demand and water supply is widening, making effective water resource management vital. Increasing competition for scarce water resources requires that the available resources be managed in a way that ensures equity while also improving economic return per unit of water available.

Improved irrigation management will be essential to sustain and improve agricultural production, and to meet increasing food demands in developing countries. In future, as competition for available water resources increases, irrigated lands will be required to use water more efficiently. Further irrigation development will be required in the long term in response to increasing food demands (Jensen et al. 1990).

In irrigation systems, it is increasingly being considered necessary to introduce water charges in order to encourage better water management practice at farmer level. This in turn requires that the system be managed in an equitable manner, and that operators have a basis for justifying their management decisions. It is also necessary to consider competing water demands, such as irrigation, potable water and hydropower, and to have a basis on which these may be compared. This may not be solely economics. Increasingly, social and environmental factors must be incorporated in the decision process (Jensen et al. 1990).

1.2 Background To The Research

As effective water management becomes more critical, water distribution in irrigation systems should be both equitable and efficient. The improved management of water within irrigation systems is essential if scarce resources are to be used in an equitable manner and to their maximum benefit. The management of water allocation in large irrigation systems is important as even a small improvement in operation may lead to significant benefits.

This research is intended to address problems of irrigation water management and in particular that of optimal and equitable distribution of irrigation water among farmers in times of water scarcity. In many irrigation systems competition for scarce water exists among farmers and/or schemes. In complex irrigation systems, the distribution of water may not be equitable or optimal leading to poor irrigation water management and sub-optimal crop production. The real time irrigation water management problem is one of providing the

best distribution of scarce water resources given a particular cropping and soil moisture condition. Decision making in real time for water allocation in irrigation systems under stress is important, as is the development of tools to assist in the assessment of different operational rules and rotational systems and to assist in evaluation of their impact on production (Wardlaw et al. 1997).

Recent studies by Wardlaw and Barnes (Wardlaw and Barnes 1996; 1997; 1998) have been concerned with the real time allocation of water supplies in irrigation systems with complex distribution networks. This work has developed a quadratic programming (QP) approach to optimise water allocation. There were concerns that the QP approach may become computationally bounded on large system and that it did not lend itself well to water scheduling problems. Genetic algorithms were identified as a possible alternative approach.

1.3 The Objectives of The Research

One of the major goals in the management and operation of an irrigation system is the efficient, distribution and utilisation of water. Operating open channel systems optimally to achieve maximum efficiency is very important. Genetic algorithms (GAs) have not yet been applied to this type of problem. The main purpose of this research is to develop an optimisation approach based on genetic algorithms (GAs) for the real time water allocation problem that maintains equity between schemes and production units within a system. The research was intended to explore the suitability of GAs to irrigation management problems. The specific objectives were as follows:

1. To develop a GA for application to the real time irrigation water allocation problem.
2. To evaluate GA performance against that of Quadratic Programming (QP).
3. To package the optimisation approach in a generic form for easy use.
4. To develop a GA for application to water scheduling problems.

1.4 Thesis Organisation

This thesis is organised as follows:

Chapter 1 outlines the general background to the research, and the objectives.

In Chapter 2, the real time water allocation problem is described and the previous work is reviewed.

Chapter 3 presents a review of the literature on optimisation approaches, including GA techniques, and their application to irrigation systems.

Chapter 4 presents the development of a GA for the water allocation problem: It begins with a single canal, then develops to application to a test canal system, and application to a more complex canal system. An appropriate objective function for GA use is developed.

In Chapter 5, application of the GA to the irrigation system of the Tukad Ayung in Bali, Indonesia is presented. The results are compared with the quadratic programming approach developed by Wardlaw and Barnes (1998). The development of the objective function for this complex irrigation system has been described. Furthermore, the development of an improved GA for water allocation problems is presented. The improved GA is compared with the original application.

Chapter 6 reviews the literature for water scheduling problems and presents an appropriate objective function for this type of problem. The application of GA to water scheduling to a test system is presented.

Chapter 7 presents the development of a GA for the water scheduling problem in a more complex system. The objective function is modified and applied to a variety test systems.

In Chapter 8, an application of a GA to the Pugal system at Indira Gandhi Nahar Project in India is demonstrated.

Chapter 9 presents the application of GA a to Hetao Irrigation Project, Mongolia. Results were compared with the original linear programming.

Chapter 10 provides the conclusions of the research. The chapter lists the achievement and outlines the limitations of the work carried out. Recommendations for further study are also provided.

2 THE REAL TIME WATER ALLOCATION PROBLEM

2.1 Introduction

This research builds on recent investigations by Wardlaw and Barnes (1996, 1997, 1998, 1999), addressing what is known as the "Real Time Water Allocation Problem". The research by Wardlaw and Barnes was undertaken as a component of the DFID TDR project entitled "Improved Irrigation System Planning and Management" (IISPM), and was concerned specifically with the optimal allocation of scarce water resources in real time between competing users. The real time irrigation water allocation problem was defined as one of "providing the best distribution of scarce water resources, such that crop yield reductions resulting from water stress are minimised."

The research by Wardlaw and Barnes was aimed at improving the availability of water for sustainable food production in irrigation networks with complex distribution networks, in which there is water stress and competition for scarce water resources. The research was intended to address issues of equity in water distribution, and to develop objective means by which the distribution of scarce water resources could be carried out in the most beneficial way. The research focussed on the development, testing and verification of computer software for optimising the allocation of water resources according to a defined set of objectives, and subject to a defined set of constraints.

The research by Wardlaw and Barnes provides the foundations for this thesis. In view of this a thorough review of their research and findings is presented in this chapter, leading to identification of the main focus of the thesis.

2.2 Definition of the Optimisation Problem

Wardlaw and Barnes (1996) defined as their objective, maximising crop production through water allocation, subject to constraints of equity between users, canal capacities, and continuity throughout the system. A number of alternative objective function formulations is possible for the water allocation problem, and were considered by Wardlaw and Barnes (1996). They found the following formulation to be most appropriate:

$$Minimise \qquad Z = \sum_{i=1}^{n} \frac{(d_i - x_i)^2}{d_i} \tag{2.1}$$

where,

n = number of irrigation schemes

d_i = irrigation demand for scheme i

x_i = irrigation supply to scheme i

The quadratic form of this function ensures equity between schemes, and that, subject to canal capacities, relative supplies are the same to all schemes. The system constraints required related to nodal continuity and to reach capacities. A node may be considered to represent a canal distribution or confluence point in the system, and for the definition sketch shown in Figure 2.1, the nodal continuity may be expressed as:

$$Qinf_i + \sum_{j=1}^{m} Q_{ij} - x_i - Qsnk_i = 0 \tag{2.2}$$

where,

$Qinf_i$ = external inflow to node i

Q_{ij} = flow from node i to node j

$Qsnk_i$ = sink or drainage outflow term at node i

The following additional constraints were required:

$$Q_{ij} \leq qcap_{ij} \tag{2.3}$$

$$x_i \leq d_i \tag{2.4}$$

where,

$qcap_{ij}$ = capacity of canal between nodes i and j.

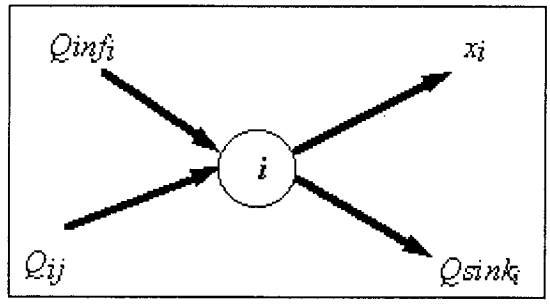

Figure 2.1 Definition sketch for nodal water balance

Wardlaw and Barnes also developed a crop yield based objective function. Starting with the objective of maximising crop yield, or minimising crop losses, while maintaining equity between schemes, it follows from equation 2.1 that a crop yield based objective function may be expressed as:

$$\textit{Minimise} \qquad Z = \sum_{i=1}^{n} \frac{1}{y_{im}} (y_{im} - y_{ia})^2 \tag{2.5}$$

where,

y_{im} = potential crop yield in scheme i

y_{ia} = actual crop yield in scheme i

It was recognised that equation 2.5 could not be applied practically, and that yield response was best related to potential and actual evapotranspiration, and through this to water supply. It was shown by Wardlaw and Barnes (1997, 1998) that through consideration of a crop yield response to water function (Doorenbos and Kassam, 1979):

$$\left(1 - \frac{y_a}{y_m}\right) = k\left(1 - \frac{ET_a}{ET_m}\right) \tag{2.6}$$

where,

k = crop yield response factor

ET_a = actual crop evapotranspiration

ET_m = potential evapotranspiration

The objective function given by equation 2.5 can be re-arranged and expressed as:

$$\textit{Minimise} \qquad Z = \sum_{i=1}^{n} \frac{k_i^2}{d_i^2} (d_i - x_i)^2 \tag{2.7}$$

or with different sensitivity:

$$\textit{Minimise} \qquad Z = \sum_{i=1}^{n} \frac{k_i}{d_i} (d_i - x_i)^2 \tag{2.8}$$

The above functions are subject to the constraints given in equations 2.2 to 2.4. Wardlaw and Barnes (1997, 1998) also extended the above functions to include blocks within an irrigation scheme, permitting two levels of optimisation:

$$\textit{Minimise} \qquad Z = \sum_{i=1}^{n} \sum_{j=1}^{b(i)} \frac{k_{ij}^2}{d_{ij}^2} (d_{ij} - x_{ij})^2 \tag{2.9}$$

or

$$Minimise \qquad Z = \sum_{i=1}^{n} \sum_{j=1}^{b(i)} \frac{k_{ij}}{d_{ij}} \left(d_{ij} - x_{ij}\right)^2 \tag{2.10}$$

where.

k_{ij} = crop yield response factor in block j of scheme i

n = number of irrigation schemes

$b(i)$ = number of irrigation blocks in scheme i

d_{ij} = irrigation demand in block j of scheme i

x_{ij} = irrigation supply to block j of scheme i

2.3 Optimisation Solvers Considered

A number of optimisation approaches were considered by Wardlaw and Barnes (1997). These included linear programming, non-linear programming, dynamic programming, evolutionary algorithms, and simulated annealing. Linear programming was not appropriate in view of the non-linear nature of the objective functions identified. Dynamic programming had been considered by Wardlaw and Coals (1994), and was shown to be extremely difficult to set up in a generic way. Evolutionary algorithms and simulated annealing were recognised as techniques that could have significant potential, but the lack of any well defined approaches within these categories led to them being put aside in favour of quadratic programming approaches. A number of commercial quadratic programming solvers is available and were also reviewed. Of these, both LINGO (LINDO Systems, 1995) and NAG Routine e04nfc (NAG, 1995) were utilised. LINGO was found to be particularly useful during model development, while the NAG routines offered better long term development potential as they could be incorporated directly into any developed executable code.

2.4 Objective Function Performance

Wardlaw and Barnes (1998) demonstrated the validity of their objective functions through application to the simple test network shown in Figure 2.2. From equation 2.6, the relative yield may be expressed as:

$$\frac{y_a}{y_m} = 1 - k\left(1 - \frac{x}{d}\right) \tag{2.11}$$

and for equity should be the same for all nodes. A series of experiments were carried out with each objective function using a range of inflows and with variations in the yield response factors k_i. The results of these tests are presented in Table 2.1. Clearly the objective function of equation 2.9 does not result in an equitable yield response, giving a strong bias to the node with the higher yield response factor. Equation 2.10 does provide an equitable yield response and is clearly the more appropriate function.

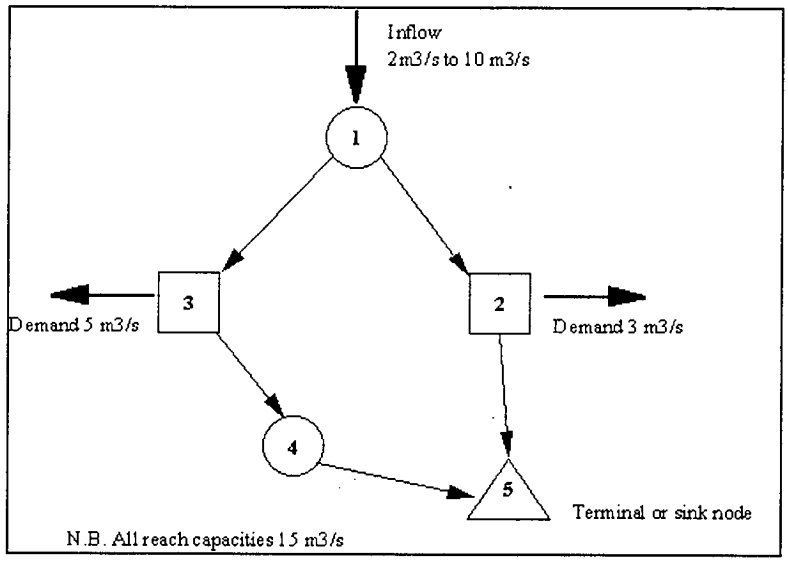

Figure 2.2 *Test network for objective function evaluation* (Wardlaw and Barnes, 1997)

Table 2.1Evaluation of crop yield based objective function for network in Figure 2.2

(after Wardlaw and Barnes 1997)

Z	$Z=\sum_{i=1}^{n}\sum_{j=1}^{b(i)}\frac{k_{ij}^2}{d_{ij}^2}(d_{ij}-x_{ij})^2$							
Node	2				3			
k_i	2				1			
Inflow	d_I	x_I	x_I/d_i	y_a/y_m	d_I	x_I	x_I/d_i	y_a/y_m
10	3	3	1	**1**	5	5	1	**1**
8	3	3	1	**1**	5	5	1	**1**
6	3	2.83	0.94	**0.89**	5	3.16	0.63	**0.63**
4	3	2.67	0.89	**0.78**	5	1.33	0.26	**0.27**
2	3	2.00	0.67	**0.33**	5	0.00	0.00	**0.00**
Z	$Z=\sum_{i=1}^{n}\sum_{j=1}^{b(i)}\frac{k_{ij}}{d_{ij}}(d_{ij}-x_{ij})^2$							
Node	2				3			
k_i	2				1			
Inflow	d_I	x_I	x_I/d_i	y_a/y_m	d_I	x_I	x_I/d_i	y_a/y_m
10	3	3	1	**1**	5	5	1	**1**
8	3	3	1	**1**	5	5	1	**1**
6	3	2.54	0.85	**0.69**	5	3.46	0.69	**0.69**
4	3	2.08	0.69	**0.38**	5	1.92	0.38	**0.38**
2	3	2.00	0.54	**0.08**	5	0.38	0.08	**0.08**
Z	$Z=\sum_{i=1}^{n}\sum_{j=1}^{b(i)}\frac{k_{ij}}{d_{ij}}(d_{ij}-x_{ij})^2$							
Node	2				3			
k_i	0.9				1.5			
Inflow	d_I	x_I	x_I/d_i	y_a/y_m	d_I	x_I	x_I/d_i	y_a/y_m
10	3	3	1	**1**	5	5	1	**1**
8	3	3	1	**1**	5	5	1	**1**
6	3	2	0.67	**0.7**	5	4	0.8	**0.7**
4	3	1	0.33	**0.4**	5	3	0.6	**0.4**
2	3	0	0	**0.1**	5	2	0.4	**0.1**

2.5 Evaluation of Optimisation Potential

2.5.1 General

Wardlaw and Barnes (1997, 1998), evaluated the performance of their optimisation approach through application to the irrigation system from the Tukad Ayung in Bali. A simulation model of this system was available (Wardlaw and Wells, 1996), and had been developed as part of a water resources investigation of Bali Province (Sir M MacDonald and Partners Asia, 1989). The simulation model was considered to provide a good representation of actual system operation, and therefore served as a benchmark against which the effectiveness of the optimisation approach could be measured.

The simulation modelling approach is described by Wardlaw and Wells (1996), and comprises the following components:

- a hydrological model of the upper catchment area
- a network distribution or systems simulation model for the irrigation system
- an in-field water balance model
- a crop production model

The hydrological model of the upper catchment area was of little relevance to evaluation of the optimisation approach, which was concerned initially with the network distribution system. The evaluation was carried out in two stages. Initially a simplified model was set up in which the water balance and crop production components of the model were removed. This was followed by a more comprehensive modelling approach in which the optimisation approach was incorporated with the water balance and yield response functions.

A schematic of the Tukad Ayung irrigation system is shown in Figure 2.3. The system is complex, and there is significant drainage re-use both within and between schemes. Drainage routes are shown dotted in Figure 2.3, and contributing drainage areas are indicated. The model schematic of the irrigation system is shown in Figure 2.4.

The simulation model was designed to run with 34 years of historical inflows, thereby permitting evaluations to be made over a wide range of conditions, and providing a reasonable base for statistical analyses of system performance.

2.5.2 Simplified Model Performance

One of the functions of the field water balance model was the determination of drainage return flows. In the models used for preliminary testing, drainage return flows were expressed as a fixed proportion of irrigation supply (Wardlaw and Barnes, 1997). Actual drainage returns are of the order of 40-50% for the system, and in preliminary model evaluation, values of 10%, 30% and 50% have been used, giving a fairly wide range over which to judge optimisation model performance. The models were driven using design irrigation demands, and ignoring effective rainfall. A much higher water stress situation was thus being imposed than in fact exists in the system.

Wardlaw and Barnes (1997) evaluated performance over the entire 34 year simulation period, using measures of drainage outflow from the system, and relative irrigation deficits. Table 2.2 summarises average system outflows from the optimisation and simulation models. As drainage returns are reduced and water stress in the system increased, particularly in the lower part of the system, then the optimisation model produces progressively greater benefits in terms of overall water utilisation. The average reduction in outflows achieved with the simulation model is shown in Figure 2.5, along with the reductions achieved in 1966 which was a particularly dry year. The benefits of optimisation are clear.

Table 2.2 Average irrigation water supplies, 1953-86 (after Wardlaw and Barnes, 1997)

Drainage Factor	Simulated Irrigation Supply (m^3/s)	Optimised Irrigation Supply (m^3/s)	Optimisation Benefit (m^3/s)
0.5	9.13	9.82	0.70
0.3	8.78	9.57	0.79
0.1	8.40	9.24	0.85

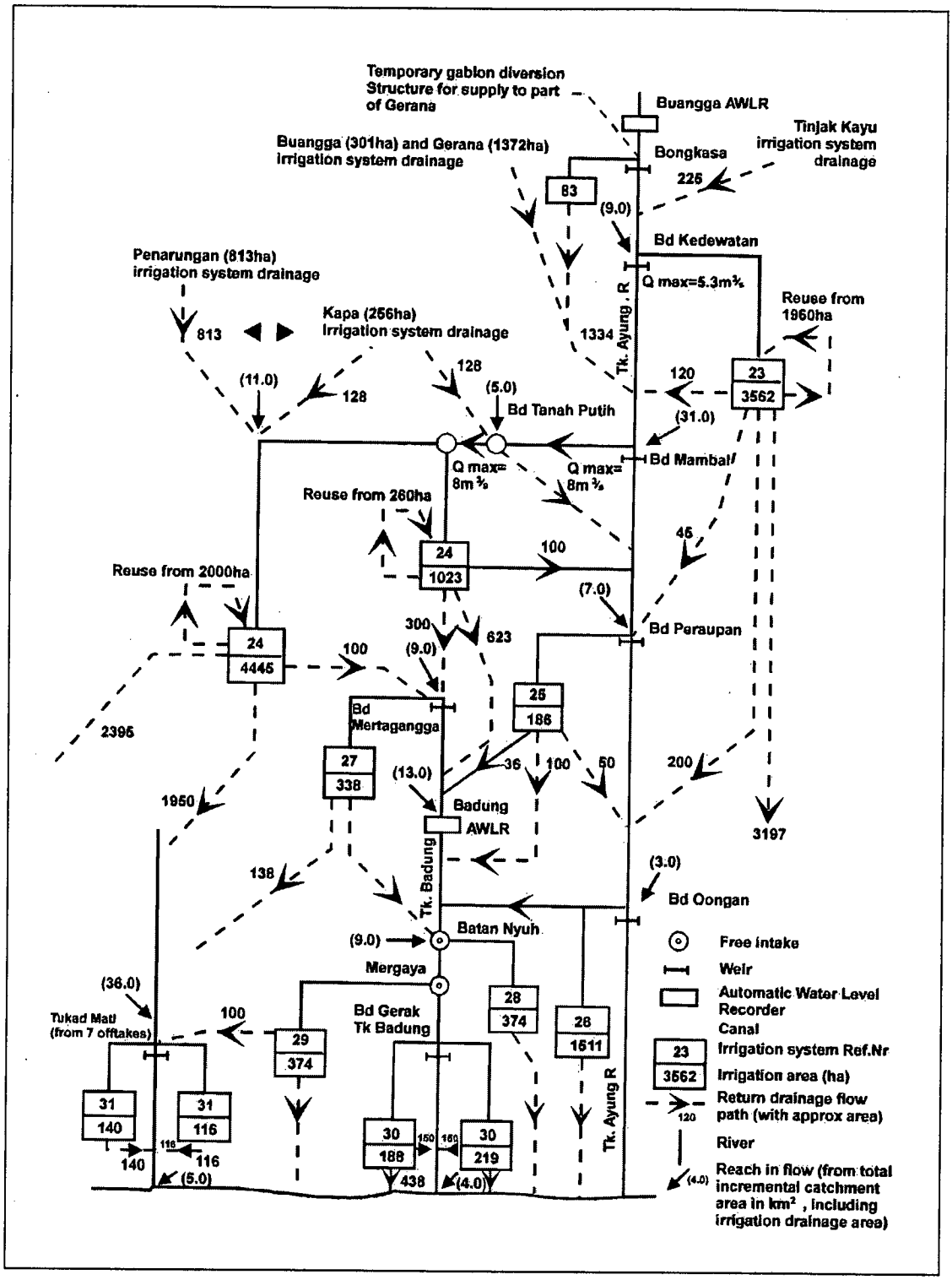

Figure 2.3 Tukad Ayung irrigation system schematic

(after Wardlaw & Barnes, 1997)

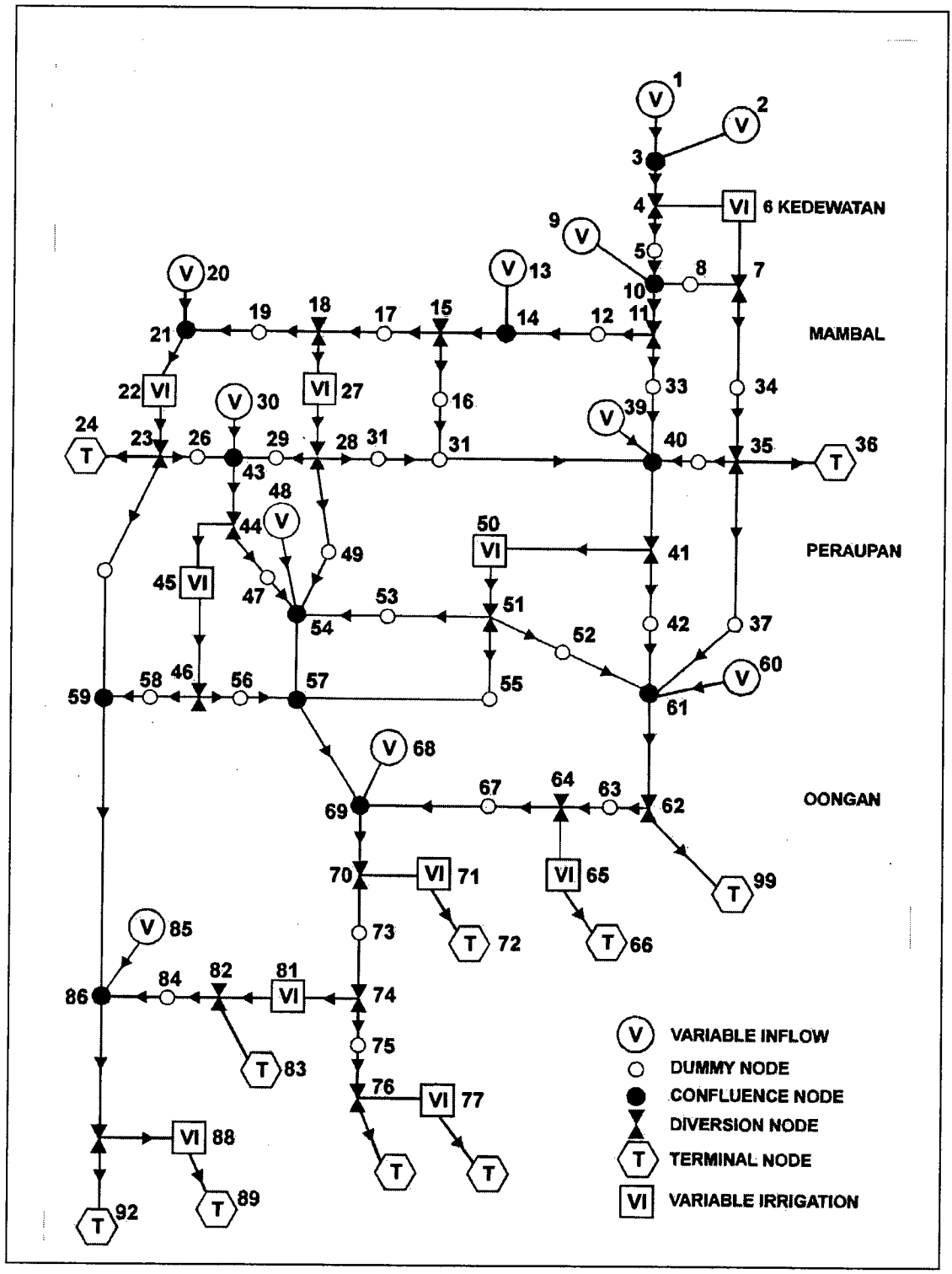

Figure 2.4 *Network diagram for the lower Ayung system simulation model*

(after Wardlaw & Barnes, 1997)

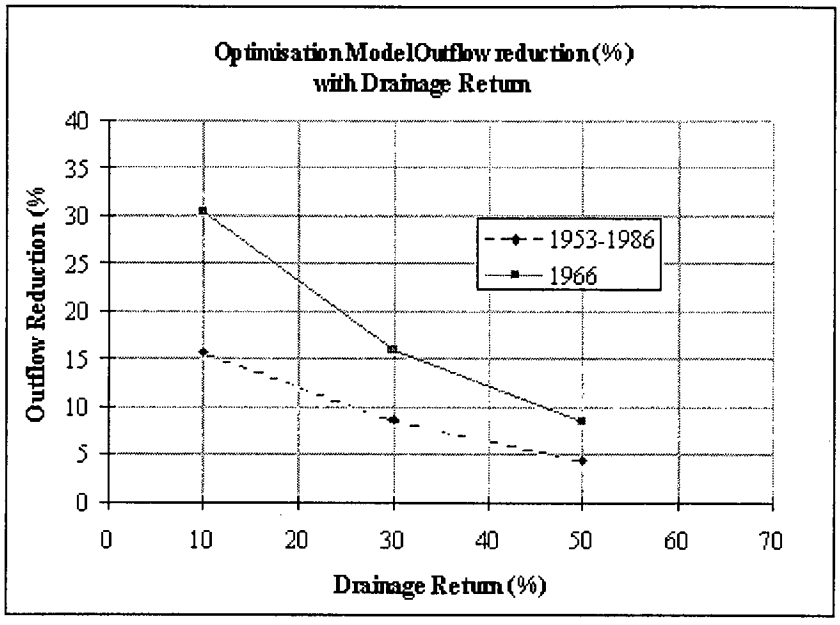

Figure 2.5 *Effectiveness of optimisation model in reducing outflows* (after Wardlaw and Barnes, 1997)

Equity in supply was an important feature of the objective functions developed by Wardlaw and Barnes. They have shown with the simplified model, that irrigation supplies were maximised in an equitable manner. Figure 2.6 shows the relative irrigation supply deficits, defined as the difference between irrigation demand and actual supply, divided by demand. It is clear from Figure 2.6 that the model achieves near equity, subject to system constraints. As a management and decision aiding tool, the optimisation approach clearly has significant potential. Irrigation deficits are reduced by about the same amount as drainage outflows with the optimisation approach, but it is in terms of equity that the real benefits lie.

On the basis of the preliminary results, Wardlaw and Barnes considered that there was sufficient justification for the development of a more sophisticated and practical approach in which account could be taken of soil moisture conditions, actual irrigation requirements and drainage returns in a more realistic manner than had been possible in the preliminary model.

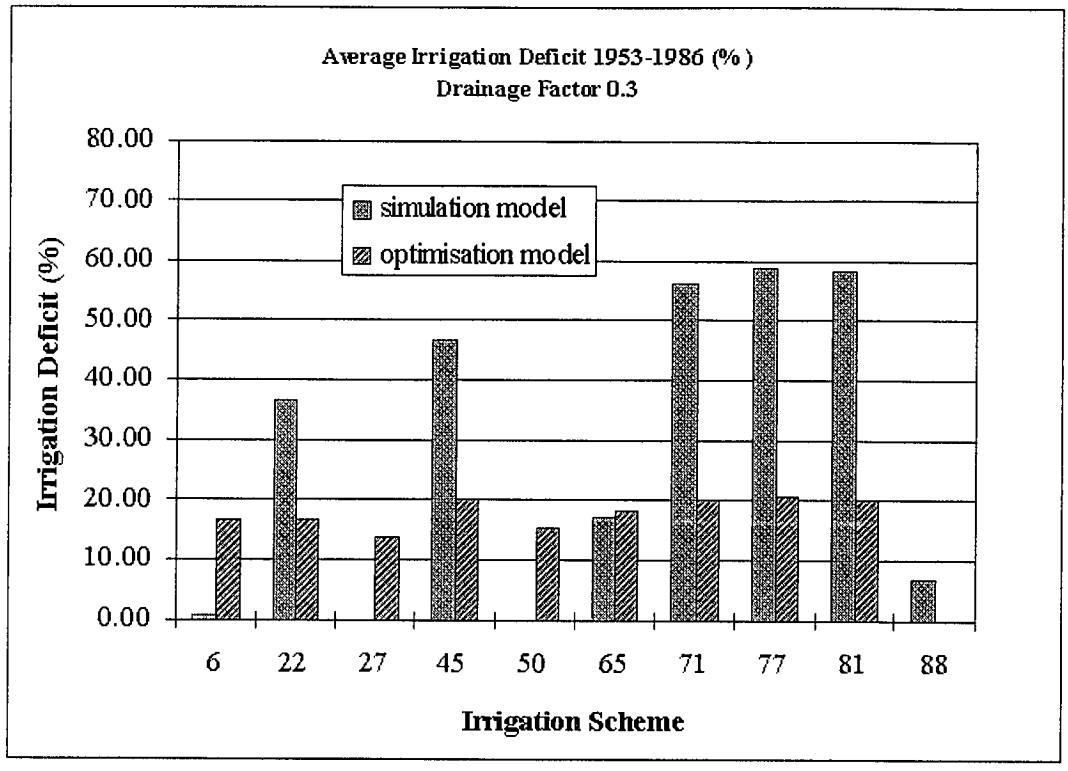

Figure 2.6 *Average relative irrigation deficits, 1953-86*
(after Wardlaw & Barnes, 1997)

2.5.3 The Integrated Model

Wardlaw and Barnes (1997) reasoned that for a more realistic assessment of optimisation performance, and for real time application, a soil moisture balance model should be incorporated into the optimisation model. Their objectives in incorporating a soil moisture balance model were to permit:

- calculation of field response to irrigation and rainfall, including surface runoff, groundwater recharge and baseflow, with feedback to the optimisation model
- determination of actual irrigation requirements
- calculation of actual and potential crop evapotranspiration, from which crop yield response could be determined

The soil moisture balance model developed was based on that used in the original Lower Ayung simulation model (Wardlaw and Wells, 1996). Significant modifications were required, however, in order to integrate the model with the optimisation model approach. In the simulation model, it had been possible to operate the systems simulation model in a

sequential manner, working from upstream to downstream, since downstream schemes had no influence on upstream schemes. In the optimisation approach it was necessary to consider all schemes concurrently since water allocation decisions are based on the state of the system as a whole. This was achieved through effectively increasing the dimensionality of the soil moisture balance model, while retaining the same process representation as in the original model. For a given set of inputs to individual schemes, the re-structured model reproduced the same results as the original model (Wardlaw and Barnes, 1998).

In the integrated model, which was called IISPMOPT, the soil moisture balance model simulates system response to water allocations determined through optimisation, computing drainage returns, actual evapotranspiration and soil moisture storage. Wardlaw and Barnes evaluated IISPMOPT though a number of experiments, using the simulation model as a benchmark. The identifiers used for model runs were as follows:

SIM	simulation model
OPT1	optimisation model with equitable water allocation objective function, and theoretical irrigation demands
OPT2	optimisation model with yield based objective function and real time irrigation demands (i.e. demands computed by soil moisture balance model)

Wardlaw and Barnes (1998) expressed the results of their experiments in terms of crop losses at different return periods. The crop production and economic models used were based on those described by Wardlaw and Wells (1996) and incorporated in the original simulation model. A comparison of annual crop losses produced by the simulation model, and by IISPMOPT, operating with the equitable water allocation function, is shown in Figure 2.7. There are significant benefits through optimisation, both in terms of equity and reduction in crop losses. Generally the optimisation model reduced the financial value of crop losses by about 50%.

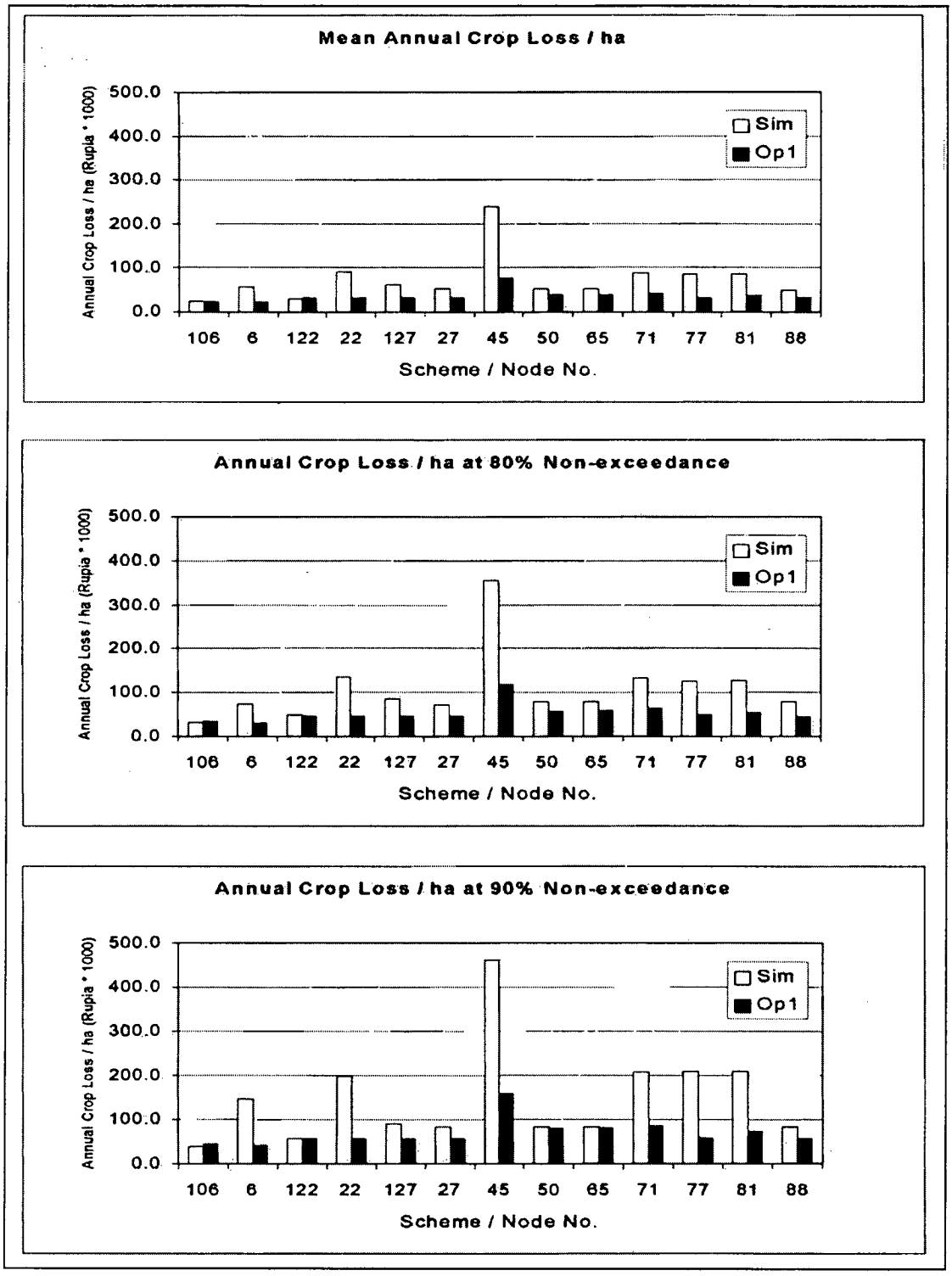

Figure 2.7 *Annual crop losses / ha, present cropping* (after Wardlaw & Barnes, 1998)

Assessment of the crop yield response objective function required that the optimisation model be run with real time irrigation demands, rather than with design irrigation demands, as had been done in the OPT1 experiments. Rather surprisingly, it was found that using real

time irrigation demands did not improve crop production from the system. The main reason for this was that in a system dominated by rice cultivation, significant field storage exists. The current practise of over irrigating when supplies permit, ensures that field storages remain fairly full, providing buffering through drought periods (Wardlaw and Barnes, 1998). Results with real time irrigation demand forecasts were more equitable than with design irrigation demands. The yield based objective function was assessed through comparison of its performance with that of the water allocation based function. The identifiers used for runs were as follows:

Opt1_f	water allocation based objective function, forecast irrigation demand
Opt2	crop yield based objective function, forecast irrigation demand

The comparison of model results using these objective functions is shown in Figure 2.8. The crop yield based objective function does not perform as well as the much simpler water allocation function. Wardlaw and Barnes (1998) were able to put forward several reasons for this. They noted that the yield response factors for dry foot crops were often higher than those of rice, resulting in a distribution bias to these crops, while their financial value was lower. This issue was addressed by introducing an economic factor also, and this redistributed the crop losses somewhat between schemes, depending upon cropping patterns. It did not improve performance relative to the water allocation function, however. Another factor was thought to be that because of bias introduced through crop yield factors, some crops would be given more than their absolute minimum irrigation requirement, at the expense of crops with lower yield response factors. It was concluded that the yield response functions did not, in the form of equations 2.7 and 2.8, provide any advantages over the simpler water allocation function, which in any case provided a concept more readily acceptable and measurable for farmers.

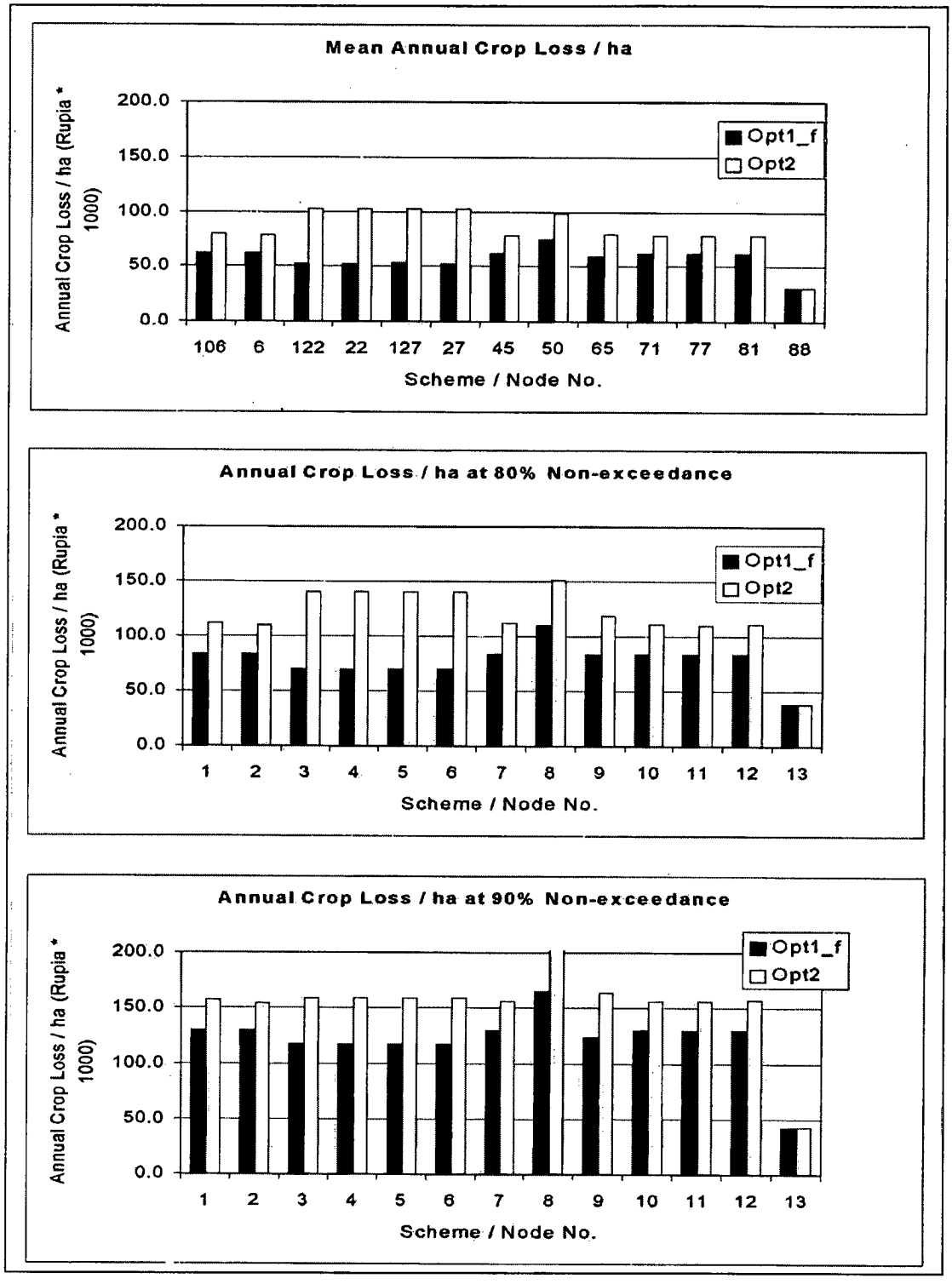

Figure 2.8 *Annual crop loss / ha, forecast irrigation demands*

(after Wardlaw & Barnes, 1998)

2.6 Summary Conclusions

Wardlaw and Barnes (1998) developed a very effective optimisation approach for real time water allocation in irrigation systems with complex distribution networks, based on quadratic programming. Through simulation experiments they demonstrated that the approach improved equity and could significantly reduce crop losses induced through water stress. They concluded that an objective function based on water allocation was, for rice cultivation at least, more effective than a more complex function based on crop yield response.

An effective user interface was developed, based on a relatively transparent matrix representation of canal and nodal connectivity. The quadratic programming approach was demonstrated to be effective, and even in application to larger networks was capable of producing optimised allocations. Areas of potential future research were identified in the investigation of alternative optimisation approaches, and evolutionary algorithms in particular, investigation of the application of optimisation approaches to water scheduling problems, and further development and investigation of the crop yield based objective functions.

This thesis addresses two of these identified areas for future research investigation of alternative optimisation approaches, and investigation of the application of optimisation approaches to water scheduling problems.

3. OPTIMISATION APPROACHES

This chapter provides a brief review of optimisation techniques, and considers their potential application to irrigation water allocation problems. Genetic Algorithms (GAs) and their applications to water resources problems are also reviewed and considered in some detail.

3.1 The Historical Context

Optimisation techniques are used in a wide range of applications in many related fields including operations research, decision science, management science, as well as in decision support for a wide range of engineering applications. Most of the techniques in use were developed to solve practical problems to which optimal solutions were required.

During the 2nd world war, British scientists and engineers were asked to analyse several military problems such as the deployment of radar and the management of convoy, bombing, antisubmarine, and mining operations. These problems were so called operations research or management science problems. The required a scientific approach to system design, operation, and decision making, normally maximizing the benefit accrued through the deployment of scarce resources (Winston 1994).

In 1947, Dantzig developed the simplex algorithm for solving linear programming problems (LP). Since then, LP has been used to solve optimisation problems in a wide range of industries, banking, education, natural resources management, and transport (Winston 1994). LP is an optimisation approach in which a linear objective function of the decision variables is maximised or minimised. The values of the decision variables must satisfy a set of constraints, each of which must be a linear equation or linear inequality. Quadratic programming (QP), one of the non-linear optimisation approaches, was used by Markowitz (1959) to determine optimal investment portfolios, and is part of the work that won him the Nobel Prize in Economics.

3.2 General Review of Optimisation Methods in Application to Irrigation Problems

A review of commonly used optimisation methods has been made, with particular reference to their application to water resources and irrigation water management problems. Methods reviewed include:

Linear Programming (LP)
Non-linear Programming (NLP)
Quadratic Programming (QP)
Dynamic Programming (DP)
Genetic Algorithms (GAs)
Simulated Annealing (SA)
Artificial Neural Networks (ANNs)

3.2.1 Linear Programming (LP)

LP has been used to solve optimisation problems in many diverse business and industrial applications. The most important step in formulating an LP is to identify the decision variables correctly. It is powerful only for linearised problems, and both the objective function and constraints must be linear. A number of commercial solvers is in existence. For example, LINDO (Linear Interactive and Discrete Optimiser) was developed by Schrage (1986) and can be used to solve linear, and quadratic programming problems. A number of subroutines are also available from the Numerical Algorithms Group (NAG, 1995).

LP was used by Hannan and Coals (1995) in application to a water allocation problem similar to that described by Wardlaw and Barnes (1998). The LP was, however, unable to preserve equity, although it did maximise water utilisation.

Anwar and Clarke (2001) have applied a mixed integer programming approach to scheduling canal irrigation, addressing a similar problem to that considered by Suryavanshi and Reddy (1986), and by Wang et al (1995). Reddy et al (1999) formulated an irrigation scheduling problem as a zero one problem, and compared results with those obtained through the application of LINDO. These applications are considered further in Chapter 9.

3.2.2 Non-linear Programming (NLP)

In many optimisation problems either the objective function is not linear and / or some of the constraints are not linear. Such an optimisation problem is called a non-linear programming problem.

NLP problems vary in complexity from simple unconstrained single variable problems to constrained multi-variable problems. NLP solution algorithms are often unable to guarantee a global optimum for the objective function, being based on derivative and gradient search techniques. Sophisticated techniques have been incorporated in most algorithms to increase the likelihood of the global optimum being found, usually at additional computational expense. This can make NLP models slow and inefficient when dealing with large constrained problems (Hales 1994). A good introduction to NLP approaches and to quadratic programming is given by Winston (1994). Quadratic programming is a particular subset of the NLP family, which is well suited to the irrigation water allocation problem, and is discussed below.

3.2.3 Quadratic Programming (QP)

QP is a type of NLP in which either the objective function or constraints are quadratic. QP has been applied by Wardlaw and Barnes (1996, 1997, 1998) to aid decision making in the real time allocation of water in irrigation systems under stress. Their overall objective was to maximise crop production through appropriate water allocation, while maintaining equity between schemes and units within the system. The output of the study demonstrated that significant improvements in crop production could be achieved through optimisation, and that equity in water allocation could be achieved subject to canal and supply system constraints. The objective functions and system constraints developed by Wardlaw and Barnes have been outlined in Chapter 2 of this thesis. They addressed a particular problem in a unique and effective way, overcoming problems previously encountered by Hannan and Coals (1995).

3.2.4 Dynamic Programming (DP)

DP was defined as "The mathematical theory of multistage decision process" (Bellman 1957). A recent review of the literature relating to dynamic programming applications in water resources (Coals and Wardlaw, 1994) indicated that little work had been done on the

application of dynamic programming (DP) approaches for the real time operation of irrigation systems. Much of the literature is concerned with planning investigations and is related to serial systems, in which there are limited numbers of state variables. Application of DP to a complex irrigation system which is non-serial results in a large number of state variables at each stage of calculation, and has inherent dimensionality problems. A number of techniques have been devised to help overcome dimensionality problems, and are discussed by Coals and Wardlaw (1994). Of the DP approaches reviewed, non serial dynamic programming (NSDP), objective space dynamic programming (OSDP), and progressive optimality (PO) were identified as having most potential in relation to the irrigation problem.

Dynamic programming is an optimisation technique which can be used to solve both linear and non-linear sequential decision making problems. A particular problem would be divided into a number of stages, typically time periods, at each of which a decision is to be made relating to water use or distribution. Associated with each stage are a number of possible states that the system could be in, depending on the decision made in that stage, and of course the starting state. Typically, the possible states of the system would relate to reservoir storage, or soil moisture storage in a downstream scheme. In application to the real time irrigation distribution problem, a different view is required of dynamic programming. Each irrigation scheme or diversion offtake in the system represents a stage, at which decisions must be made with regard to the amount of water diverted in order to maximise crop production. The state variable associated with each stage is the amount of water supplied to the scheme, and this is of course related to the possible states in all other schemes. The number of possible combinations of states in different schemes can become very large in complex systems.

Wardlaw and Coals (1994) developed a discrete dynamic programming model for a simple canal system. They found that even with a small system, computer memory problems were difficult to avoid, and concluded that in larger systems, computational efficiency would be a major problem. Given the potentially large number of possible allocations that could be made at each stage, or decision point, a method for reducing the range of the state variable was sought. A number of successive approximation algorithms have been developed, where initial estimates of the optimal solution must be supplied by the user. New policies are successfully constructed according to the algorithm until some convergence criterion is satisfied. This means that a number of iterations are required, whereas discrete dynamic

programming (DDP) is applied only once. Examples of successive approximation techniques include discrete differential dynamic programming (DDDP) (Heidari et al, 1971), incremental dynamic programming (IDP) (Turgeon, 1982), and differential dynamic programming (Murray and Yakovitz, 1979). A detailed description of these methods and their advantages and shortcomings has been given by Coals and Wardlaw (1994). A review of these methods led Wardlaw and Coals (1994) to the development of an approach that could also be described as a successive approximation technique. The new algorithm is most similar to discrete differential dynamic programming (DDDP) in that it provides an initial feasible solution and then uses increments to move towards an optimal solution. This reduces the number of possible states to be considered. It differs from DDDP in that the increments are not specified beforehand (and therefore no corridors are defined) but are set at the beginning of each iteration by the user, thus allowing for out of model decision making.

The approach has been applied to an irrigation allocation problem. For small scale systems with no inter basin transfer via diversions, this algorithm can solve the allocation problem very simply, and explicitly, by applying a factor representing the ratio of supply to demand to each of the stage demands. For more complex systems it was found that the approach was very cumbersome, and could not be set up in a transparent manner. Wardlaw and Coals concluded that a QP approach would afford a more robust approach to the water allocation problem, permitting a more complete description of the problem.

Wardlaw and Barnes (1997) tested and compared a dynamic programming approach with the quadratic programming for water allocation on part of the South Lombok irrigation system in Indonesia. The DP approach was not generic and the number of possible combinations of states in different schemes could become very large in complex systems. No future was seen for DP based approaches to the irrigation water allocation problem.

3.2.5 Genetic Algorithms (GAs)

A GA is search algorithm based upon the mechanics of natural selection, derived from the theory of natural evolution – hence the term evolutionary algorithms. GAs simulate mechanisms of population genetics and natural rules of survival in pursuit of ideas of adaptation. The study of GAs originated in the mid 1970s (Holland, 1975). They have become a powerful optimisation approach, although as yet have seen limited application in

water resources. Excellent introductions to GAs are given by Goldberg (1989) and by Michalewicz (1992), and several recent papers give summaries of the essentials (e.g., Oliveira and Louks 1997, Savic and Walters 1997, and Wardlaw and Sharif 1999).

Using BIDS (Science Citation Index), an assessment has been made of the growth in the application of GAs reported in the literature. A search on "Genetic Algorithm(s)" indicates an increase from 1 article in 1986 to 689 articles in 1998. The distribution of "hits" is shown in Figure 3.1. Clearly there has been an explosive increase in applications since the mid 1990's.

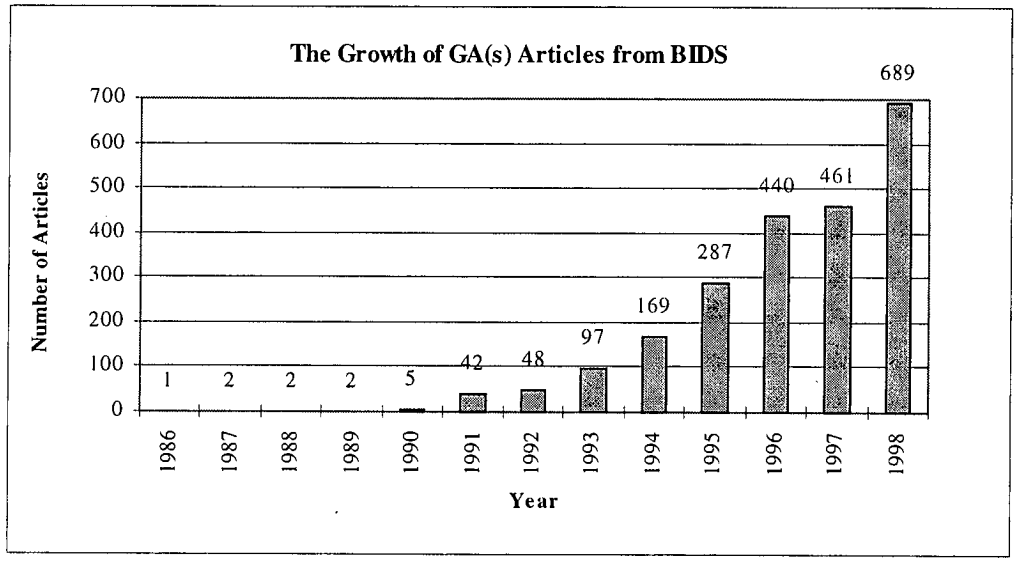

Figure 3.1 The growth of GA(s) articles from BIDS, Science Citation Index

When the GA topics were searched selectively using BIDS, only 17 articles were found that were directly related to water resources. These related to multi-reservoir systems, aquifer management, stream order, leakage losses in water distribution networks, reservoir modeling, groundwater monitoring, pipe networks, groundwater management, groundwater pollution, river and channel flood routing, piping systems, peak flow forecasting, groundwater remediation, least-cost design of water distribution, planning of storm systems, water network rehabilitation, shallow groundwater, and hydrological model calibration.

A review of GA work related to irrigation water distribution systems identified only one previous application (Chen 1997). The objective of that study was to maximize the economic benefit to 6 irrigation districts in the Shijing Irrigation Network, an important agricultural area in China. The area experiences prolonged droughts, and the objective of the

investigation was to optimise the schedule of reservoir releases to maximize economic benefits. The objective function used was not particularly complex, but included a wide range of variables and functions that would have been difficult to quantify. The study found that the conventional GA approach was not appropriate because of the large number of variables used. Modifications to the conventional GA approach were required to improve the rate of convergence.

Wardlaw and Sharif (1999) present a good review of the literature on GA applications to water resources. They concluded that little practical guidance existed on the application of GAs to water resources problems, and presented an assessment of the merits and demerits of alternative formulations in application to reservoir systems operations. From the review carried out as part of this thesis, it became apparent that GAs could have potential in application to the irrigation water allocation problem, and to irrigation water scheduling problems. The basis of the GA approach is presented in section 3.3.

3.2.6 Simulated Annealing (SA)

SA is analogous with the way that liquids freeze and crystallize, or that metals cool and anneal. The technique has attracted significant attention as a suitable optimisation tool for problems of large scale where a desired global extremism is hidden among many local extrema. It is relatively new and applications are increasing (Beasley and Heitkotter 1995). Using the BIDS Science Citation Index to search on "Simulated Annealing" with "Water management" and with "Irrigation" between 1981 to 1999 found only one article of Wang and Zheng (1998). This was related to ground water management. GA and SA are coupled with MODFLOW, a groundwater flow simulation code for optimal management of groundwater resources under general conditions. The models allow for multiple management periods in which optimal pumping rates vary with time to reflect the changing flow conditions. The objective function was incorporating multiple cost terms such as the drilling cost, the installation cost, and the pumping cost. The strengths and limitations of GA and SA based models were evaluated by comparing the results with those obtained using linear programming, nonlinear programming, and differential dynamic programming. The result show that the GA and SA based models yield nearly identical or better solutions than the various programming methods. SA tends to outperform GA in terms of the number of forward simulations needed, it used more empirical control parameters which have significant impact on solution efficiency but were difficult to determine.

3.2.7 Artificial Neural Networks (ANNs)

Neural networks have evolved from the development of pattern recognition systems from remotely sensed data. They are in effect artificial intelligence devices, which can be trained to recognise certain patterns in an input, then respond in a trained manner in producing an output from that input (Jansson 1994). Using the BIDS Science Citation Index, few articles were found when a search was conducted for "Neural Network" with "Irrigation", and with "Water Management". Brief reviews of those that were found are given below.

Atiya et al. (1999) applied ANNs for river flow forecasting. The objective of that study was to apply ANNs to the problem of forecasting River Nile flows in Egypt. The time series of historic flows was used as a benchmark to compare several neural-network forecasting methods.

Yang et al. (1997) developed ANNs to simulate fluctions in water-table depths and drain outflows for subsurface drained farmlands in Ontario. The problem was one of water table control to ensure adequate drainage, while providing subsurface irrigation also. Compared with conventional models, the ANNs model required fewer inputs parameters, and had much faster execution times. These benefits can be exploited to good effect in the real-time control of water-table management systems. Such models require to be trained, however. Yang et al. (1996) trained their model using simulated water table depths from DRAINMOD, a conventional water table management model. Compared to DRAINMOD, the model was very simple to run, and required only precipitation, evapotranspiration, and initial water-table depths to run.

No ANN applications have been found involving optimisation of water management in open channel or canal systems.

3.2.8 Combined Methods

Hybrid methods can take advantage in various optimisation problems, and a brief review of the potential of such approaches, with non-particularly reference to water resources and water management problems is also outlined below.

Wang et al. (1999) combined Monte Carlo simulations with a GA to both sample the conformational spaces and through this search the global conformations of peptides. The

results indicated that this procedure could successfully explore a set of the conformational spaces and could also find the global conformations for most peptides.

Huang and Arora (1997) studied optimal design with discrete variables. Continuous-discrete variable non-linear optimisation problems were defined and categorized into six different types. These included a full range of problems form continuous to purely discrete and non-differentiable. Methods for solution of these problems were studied and their characteristics were catalogued. Branch and bound, SA, and GA were found to be the most general methods for solving discrete problems. To study performance of the methods, several example problems were solved. It was found that solution of the mixed variable non-linear optimization problems usually required considerable more computational effort compared to the continuous variable optimization problems. However, there was no conclusion that the best solution had been obtained, but good practical solutions are usually obtained.

Cieniawski et al. (1995) used GAs to solve a multi-objective groundwater monitoring problem of optimal location of network of groundwater monitoring wells under uncertainty condition. GAs were allowed to consider the two objectives, maximizing reliability and minimizing contaminated area. The GA-based solution method has the advantage of being able to generate both convex and non-convex points of the trade-off curve, accommodate non-linearities in the two objective functions, and not be restricted by the peculiarities of a weighted objective function. Furthermore, GAs have the ability to generate large portions of trade-off curve in a single iteration and may be more efficient than methods that generate only a single point at a time. The GA were compared with each other four difference coding and also with SA on both performance and computational intensity. SA relies on a weighted objective function which can find only a single point along the trade-off curve for each iteration, while all of the multiple-objective GA formulations are able to find a larger number of convex and non-convex points of trade-off curve in a single iteration. Each iteration of SA is approximately five times faster than an iteration of the GA, but several SA iterations are required to generate a trade-off curve. GAs are able to find a larger number of non-dominated points on the trade-off curve, while SA finds fewer points but with a wider range of objective function values. None of the GA formulations demonstrated the ability to generate the entire trade-off curve in a single iteration. Through manipulation of GA parameters certain sections of the trade-off curve can be targeted for better performance, but as performance improves at one section it suffers at another.

From the review of hybrid methods, it would appear that combination methods are alternatives for further studies.

3.3 Genetic Algorithms

3.3.1 Basis of Genetic Algorithms

Genetic Algorithms (GAs) were first introduced by John Holland in 1971 (Holland, 1971). In 1975 Holland published the book titled "Adaptation in natural and artificial systems"(Holland, 1975), which for many years was a standard reference. Since then, GAs have been applied to a wide range of problems. A GA is an optimisation search technique based on the mechanisms of natural selection (Darwin, 1959). GA techniques are different from other optimisation techniques in a number of significant ways (Goldberg, 1989):

- GAs work with a coding of the parameter set, and not with the parameters themselves
- GAs work from a population of points, and not a single point
- GAs use objective function information, and not derivatives of other auxiliary knowledge
- GAs use probabilistic transition rules, and not deterministic rules.

In real world optimisation problems, the search space may include discontinuities and may often include a number of sub-optimum peaks. These can cause difficulties for calculus-based and enumerative schemes. A GA, is a random search algorithm. It is a robust method for searching the optimum solution to complex problems, even though it may not always necessarily lead to the best possible solution (Michalewizc, 1992). GAs search from a rich database of points or a population of strings, climbing many in the search space simultaneously. The probability of finding a peak in methods that go from one point to another point (Goldberg, 1989) is much lower, and a GA is less likely to get trapped in a local optima than other search techniques. In a GA, the problem is represented by a population of strings (or chromosomes in the biological terminology). Each string comprises a number of blocks (Figure 3.2) which represent the individual decision variables of the problem (genes in biological terminology). In early GAs (Goldberg and Kuo, 1987; Wang, 1991), the genes were comprised of binary bits (alleles in biological terminology) which

encoded the decision variables. The bits could encode integers, real values, sets, or whatever else was appropriate to the problem.

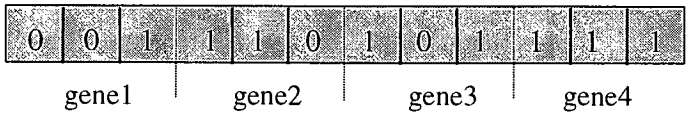

Figure 3.2 Sample of a chromosome comprises 4 genes, each gene has three alleles

Each chromosome or string represents a potential solution to the problem, and coding components of the possible solutions into a chromosome is the first part of a GA formulation. The fitness of a chromosome as a candidate solution to a problem is an expression of the value of the objective function represented by it. It is also a function of the problem constraints, and may be modified through the introduction of penalties when constraints are not satisfied.

The population of strings (or chromosomes) are processed and combined through a series of genetic operators according to their fitness, in an attempt to generate new strings combining the best features of two parent strings. The genetic operators used in the reproductive process are selection, crossover and mutation. Strings with the highest fitness have the greater chance of contributing to future generations, as in the process of natural selection (Darwin, 1959). Combination is achieved through the crossover of pieces of genetic material between selected chromosomes. Mutation allows for the random mutations of bits of information individual genes. Through successive generations, fitness should progressively improve, and at the end of the evolution process, a chromosome representing an optimal (or near optimal) solution should be obtained.

3.3.2 Representation Schemes

In a GA, an appropriate genetic representation for potential solutions to a particular problem must be defined. The choice of an appropriate coding scheme to represent the potential solutions is the key to success for any GA. The conventional representation scheme used in GAs is binary coding (Holland 1975; De Jong 1975; Goldberg and Kuo 1987). A string of bits can encode integers or real values, whichever is appropriate to a particular problem. Real value coding is now gaining popularity. Wardlaw and Sharif (1999) have shown that it has advantages over binary representation in application to reservoir problems. The

representation schemes in common use are binary, gray, and real value representation. Each of these are briefly discussed below.

a) Binary Representation

Traditionally GAs have used binary coding, in which a chromosome is represented by a string of binary bits. Binary strings are easy to operate on, and within any gene, binary representations can be mapped to values in the range which is feasible for the variable represented (Goldberg, 1989). A simple example of binary coding is shown in Table 3.1. If greater precision is required in the decimal values represented, then additional bits must be introduced to a gene. One of the drawbacks of binary coding is that mutation and crossover operations can result in significant disturbances to the values of individual genes. In Table 3.1 for example, a mutation from 111 to 011 results in the gene value changing from 7 to 3. This can lead to convergence problems. Coding and decoding of binary strings can also slow down execution time, particularly in large problems.

b) Gray Representation

Gray coding was introduced to try and reduce the impact of bitwise operations on gene values (Hollstein, 1971, Michalewicz ,1992). In gray coding, the binary value of two adjacent variable values varies by only one binary digit. Table 3.1 shows the correspondence between coded substrings for binary and Gray coding. In Gray coding, only one bit changes between adjacent substrings, and this reduces the risk of large changes in gene values between generations. In a Gray coded string, the bitwise mutation operator causes less disruption to the solution because the effect of flipping a single bit is smaller most of the time.

Table 3.1 Sample of adjacent decoding in binary and gray coding

Binary Coding	Gray Coding	Decimal Value
000	000	0
001	001	1
010	011	2
011	010	3
100	110	4
101	111	5
110	101	6
111	100	7

c) Real-value Representation

An alternative approach to formulation of the GA is to use a representation appropriate to the components of the problem. A problem specific representation gives more coherence and efficiency to the algorithm. For problems where the function arguments are integer of real values, integer or real-value representation may be used. With this approach, points that are close in the problem space are close in the representation space also, and coding or decoding is required. A significant advantage is that discritisation of the decision variable space is not required, and greater precision is therefore possible. The precision of binary coding can be improved by introducing more bits to the string, but this has the effect of slowing down execution time, and also making convergence more difficult. Michalewizc (1992) indicates that for real value optimisation problems, real-value representation out-performs binary representation because it is more consistent, more precise, and leads to faster execution. Oliveira and Louks (1997) and Wardlaw and Sharif (1999) have applied real-value representation to water resources problems.

3.3.3 The Selection Operator

The selection operator is that through which strings are selected for inclusion in the reproduction process and for participation in the next generation. The fittest strings have the highest probability of being used in reproduction. There are a number of approaches to

selection, all of which determine the probability of selection as a function of fitness. The common approaches are:

- Proportional selection
- Rank selection
- Tournament selection

a) Proportional Selection

Proportional selection is probably the most commonly used and is well described by Goldberg (1989). In proportional selection, the probability of selection (P_i) of a given string, or chromosome, is calculated by dividing the fitness of that chromosome by the summation of the fitness of all chromosomes in that generation (equation 3.1).

$$P_i = \frac{f_i}{\sum_{i=1}^{n} f_i} \qquad (3.1)$$

where;

f_i = fitness of individual chromosome(i) in that generation.

n = population size

The fitness proportional selection scheme may lead to premature convergence, however, due to high "selective pressure", which influences the degree to which the best individuals are favoured. Whitley (1989) has identified population diversity and selective pressure as two important factors influencing the genetic search process. The diversity in the population depends upon the selective pressure – a high selective pressure may lead to a rapid convergence, often to a sub-optimal point; a low selective pressure may extend the time and number of generations required to reach an acceptable solution. Yang and Soh (1997) present an interesting discussion on these points.

b) Rank Selection

Rank selection schemes operate by sorting the population according to fitness, and then assigning a probability of selection to individual chromosomes based upon their rank. A constant selection differential is maintained between the worst and the best individuals, and

information on the relative fitness of individuals in the population is ignored. Various rank selection schemes are in use (Michalewicz, 1992).

c) Tournament Selection

In tournament selection, two or more chromosomes are randomly picked from the population, and the best one selected for further genetic processing. This approach is repeated an appropriate number of times to obtain the required number of chromosomes to start off the next generation. The approach had originally been developed with groups of two individuals and was called binary tournament selection, but it has been found that using larger groups leads to greater diversity and a smoother progression to a solution. Goldberg and Deb (1991) have compared various selection schemes, and indicated a preference for tournament selection, observing that it eliminates random noise from the selection process and generally improves the efficiency of the GA search algorithm. Modifications can be introduced that ensure that the fittest chromosome is always taken into the next generation this is called the elitist strategy (Davis, 1991).

3.3.4 The Crossover Operator

Crossover is an extremely important part of a GA. If crossover is neglected, the result is no longer a genetic algorithm and the performance will be degraded for a variety of reasons (Davis 1991). The crossover operator permits the exchange of genes between pairs of chromosomes in a population. This operator offers the possibility of good genetic material from different individual strings being combined to create an even fitter individual. The operator is intended to preserve the best material from two parent strings.

The number of strings in which material is exchanged is controlled by the crossover probability, which is an input to a GA. The crossover probability is normally in the range of 0.5 and 1.0. Three approaches to crossover are described by Goldberg (1989), and by Michalewicz (1992) as:

i) one-point crossover;

ii) two-point crossover;

iii) uniform crossover.

Figure 3.3 gives an indication of how the above approaches operate. In one-point crossover, genetic material is simply exchanged about some point c, which is randomly selected. In

two-point crossover, genetic material between positions chosen at random along the chromosome length is exchanged. Uniform crossover operates on individual genes of the selected chromosomes, rather than on blocks of genes, with each gene being considered in turn for possible crossover or exchange. The use of uniform crossover leads to greater population diversity

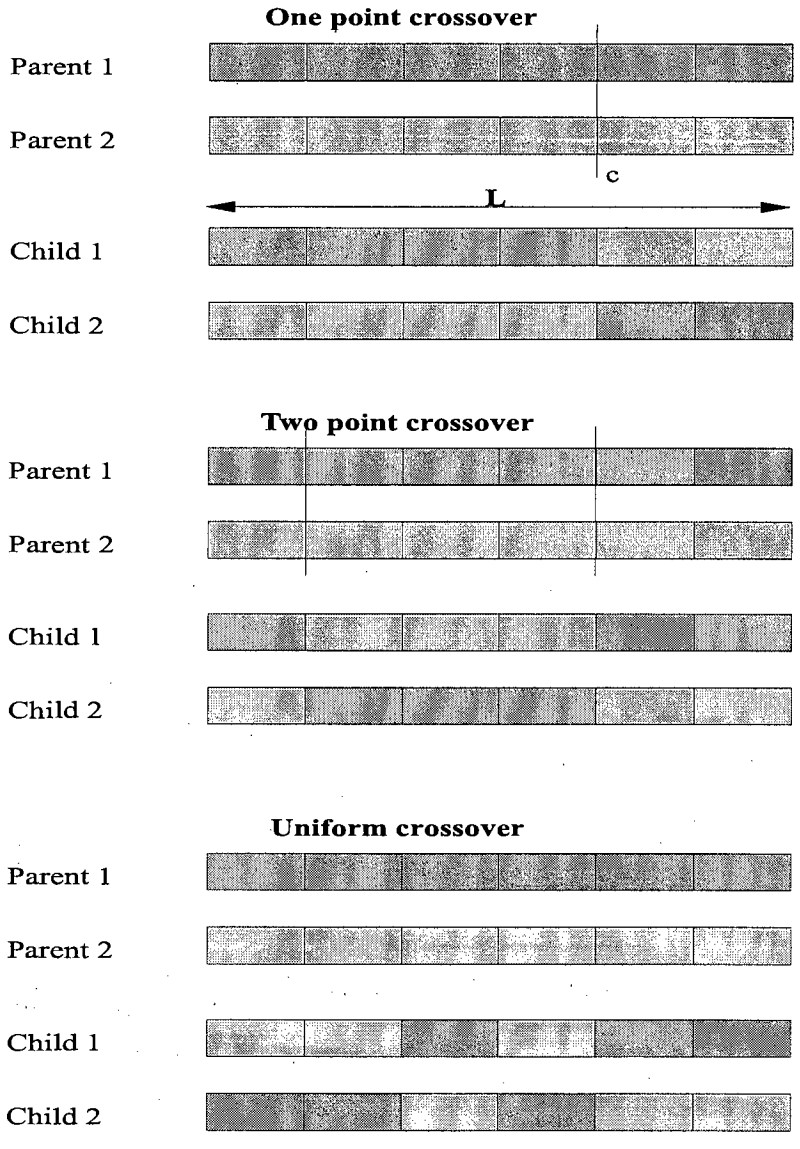

Figure 3.3 Approaches to crossover (after Wardlaw and Sharif 1999)

3.3.5 The Mutation Operator

Mutation is an important operator of GAs, and permits new genetic material to be introduced to a population. A mutation probability can be specified that permits random mutation. In binary representation, individual alleles or bits of a gene simply have their value changed from a 0 to 1 or vice versa. For real value representation Michalewicz (1992) has outlined two basic approaches to mutation. These are uniform mutation and nonuniform mutation. In uniform mutation, the value of a gene can be mutated randomly within its feasible range of values, while modified uniform mutation permits modifications of a gene by a specified amount. One of the problems with uniform mutation in real value representation is that otherwise good solutions can be disturbed by the mutation process. This can be overcome through non-uniform mutation, in which the amount by which genes can be mutated is reduced as the run progresses.

4. DEVELOPMENT OF A GA FOR AN IRRIGATION CANAL SYSTEM

4.1 Introduction

The development of a GA for the water allocation problem began with consideration of a series of relatively simple systems, on which the sensitivity of various GA approaches could be tested. These preliminary tests and sensitivities of the solution procedures to adopted parameter values are considered in this Chapter.

4.2 Alternative GA Formulations

The simple irrigation canal system shown in Figure 4.1 can be used to demonstrate how the decision variables are represented by a GA. There are four demands from the system, represented by d_1, d_2, d_3, and d_4. The inflow to the system q_1 is considered to be known, but the canal flows q_2, q_3, and q_4 and the irrigation supplies $x1$, $x2$, $x3$, and $x4$ are variables.

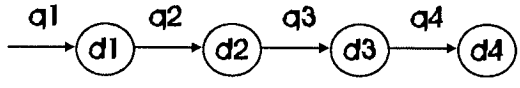

Figure 4.1 A single canal

Given the objective of maximising the allocation of irrigation water supplies in an equitable manner, the objective function may be written as in equation 2.1:

$$\textit{Minimise} \qquad Z = \sum_{i=1}^{n} \frac{(d_i - x_i)^2}{d_i} \tag{4.1}$$

Equation 4.1 may be considered as the evaluation function in GA terminology. The decision variables with respect to equation 4.1 are clearly the irrigation supplies x_i, however, these may be evaluated either directly or indirectly. The following approaches are possible:

i) use the canal flows q_2,, q_3,, and q_4 as the decision variables and determine the irrigation supplies x_1, x_2, x_3, and x_4 from the nodal water balance;

ii) use the canal flows q_2, q_3, and q_4 and the irrigation supplies x_1, x_2, x_3, and x_4 as the decision variables in the same chromosome;

iii) use the irrigation supplies *x1*, *x2*, *x3*, and *x4* as the decision variables.

The latter approach appears attractive, but in application to complex systems may leave no means of closing the water balance at certain nodes where demands do not exist. Some of the constraints would become indeterminate. Approaches i) and ii) are preferable. Considering approach i), a chromosome for the simple canal system would comprise three genes, representing q_2, q_3, and q_4. In binary representation, each gene would be comprised of a series of alleles as shown in Figure 4.2.

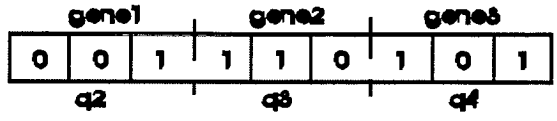

Figure 4.2 Example of a chromosome representing canal flows

A population of such chromosomes would be set up and then manipulated by the genetic operators outlined in Chapter 3. The evaluation function variables *x1*, *x2*, *x3*, and *x4* would be determined from the nodal water balance.

The following sections demonstrate application to a series of test systems through which evaluation of the alternative formulations is made.

4.3 Restatement of the Objective Function for Water Allocation

An appropriate objective function and set of constraints for the water allocation problem have been outlined in Chapter 2. These are restated here for completeness in the definition of the problem for solution using a GA. The objective function that preserves equity may be written as:

$$\textit{Minimise } Z = \sum_{i=1}^{n} \frac{(d_i - x_i)^2}{d_i} \qquad (4.2)$$

where,

d_i = irrigation demand for scheme i

x_i = irrigation supply to scheme i

n = number of irrigation schemes

Referring to Figure 2.1, the constraints to be satisfied were given in equation 2.2 to 2.4 and are:

i) capacity constraint:

$$Q_{ij} \leq qmax_{ij} \tag{4.3}$$

where,

Q_{ij} = flow to node j from node i

$qmax_{ij}$ = maximum capacity of canal connecting nodes i and j (or reach capacity)

ii) nodal balance constraint:

$$Qinf_i + \sum_{j=1}^{m} Q_{ij} - x_i - Qsnk_i = 0 \tag{4.4}$$

where,

$Qinf_i$ = external inflow to node i

Q_{ij} = flow from node i to node j

$Qsnk_i$ = sink or drainage outflow term at node i

iii) supply constraint:

$$x_i \leq d_i \tag{4.5}$$

4.3.1 Penalty Functions and Penalty Factor

In LP or NLP, constraints define the decision space and are an integral part of the solution procedure. In a GA, it is necessary to introduce penalty functions to ensure that the system constraints are satisfied. In effect, the requirement is to demote or exclude chromosomes that do not satisfy the system constraints. The system constraints given in equations 4.3 to 4.5 must be expressed in penalty function form, and the penalties are incorporated in the evaluation function. The penalty functions required for the water allocation problem are outlined below:

i) *If* $Q_{ij} > qmax_{ij}$

$$P_1 = \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{(Q_{ij} - qmax_{ij})}{qmax_{ij}} \quad (4.6)$$

where,

n = number of nodes in the system

ii) *If the absolute nodal balance > 0.001*

$$P_2 = \sum_{i=1}^{n}\frac{|Qinf_i + \sum_{j=1}^{n} Q_{ij} - x_i - Qsink_i|}{Qinf_i + \sum_{j=1}^{r} Q_{ij(in)}} \quad (4.7)$$

where,

$Q_{ij(in)}$ = inflow(s) to node i

r = number of reaches providing inflow to node i

iii) *If* $x_i > d_i$

$$P_3 = \sum_{i=1}^{n}\frac{(x_i - d_i)}{d_i} \quad (4.8)$$

where,

n = number of irrigation schemes

The objective function or evaluation function may be rewritten as :

$$Minimise\ Z = \sum_{i=1}^{n}\frac{(d_i - x_i)^2}{d_i} + R_1 \cdot P_1 + R_2 \cdot P_2 + R_3 \cdot P_3 \quad (4.9)$$

The parameters R1, R2 and R3 are weighting factors that can be used to adjust the sensitivity of the solution to different penalties. The penalty functions can also be used in quadratic form, and this increases sensitivity of the solution to their violation.

4.4 Testing on a Simple Network

Preliminary development of the GA approach was carried out on the simple network showing in Figure 4.3. The maximum capacity in each reach was set at 15 m^3/s. Initially the inflow (*Qinf*) was set to 15 m^3/s, and the demands *d1, d2, d3, d4, d5* and *d6* to 0, 4, 5, 5, 0,

and 3 m^3/s respectively. This resulted in water stress and the GA was required to distribute available resources equitably.

Network definition and inputs to the GA were handled in the same format as developed by Wardlaw and Barnes (1996) see Appendix A. Their approach offers a transparent way of defining nodes and links (canals) for the modeling system through sets of matrices. The model uses three input files:

a) a network file: to define the canal system and locations of inflows, sinks and demands

b) an irrigation scheme demands file: to define the irrigation demand in each identified demand node

c) an inflow file: to define the inflows to each identified inflow point in the system

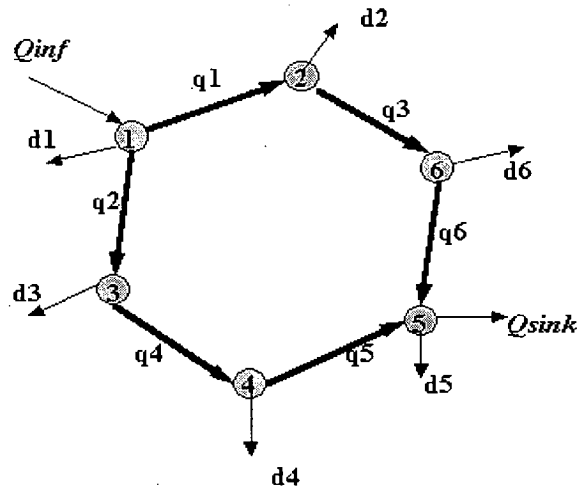

Figure 4.3 A simple canal system

4.4.1 Representation Schemes

A comparison has been made between the performance of binary and real value coding schemes on the above network. The GA was set up with canal flows as the decision variables represented in the chromosome.

a) Binary Coding Setup

An example of a chromosome representing a set of the flow parameters is shown in Figure 4.4. Each string has six genes (for six canal flows) and each gene comprises of a number of alleles or bits. These alleles are encoded/decoded by linear mapping to their decimal equivalent reach flows.

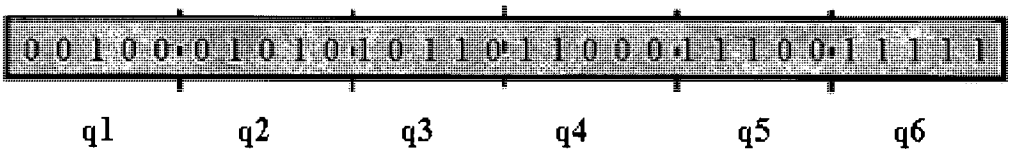

Figure 4.4 A chromosome represents the flows (q_i) in each reach.

The variables are the canal flows, and their possible range is from zero to the maximum capacity in each canal. In this problem, all canals had a maximum capacity of 15 m^3/s. An initial population of strings of canal flows was created by generating canal flows randomly within the range of 0-15 m^3/s, irrespective of continuity constraints. In binary coding, linear mapping is used to map gene values to binary codes. This is achieved as follows:

$$G_{val} = Q_{\min} + \frac{(Q_{\max} - Q_{\min})}{E_{\max}} . E_{val} \qquad (4.10)$$

where,

G_{val} = the canal flow represented by the gene

Q_{min} = the minimum possible canal flow

Q_{max} = canal capacity

E_{max} = maximum possible decimal value of gene

E_{val} = encoded decimal gene value

Table 4.1 gives examples of gene mapping for the chromosome shown in Figure 4.4.

The gene value mapping results in canal flow increments of 0.484 m^3/s. This is very poor precision, and could only be improved by increasing the length of a gene. If the gene length were changed from 5 bits to 8 bits, the maximum encoded decimal value increases to 255, and the mapped canal flow increment is reduced to 0.059 m^3/s.

Table 4.1 Mapping of gene values to variable values.

Binary Code	Encoded decimal value E_{val}	Mapped canal flow* G_{val}
0 0 0 0 0	0	0 + ((15-0)/31)*0 = 0.0
0 0 1 0 0	4	0 + ((15-0)/31)*4 = 1.9
0 1 0 1 0	10	0 + ((15-0)/31)*10 = 4.8
1 0 1 1 0	22	0 + ((15-0)/31)*22 = 10.7
1 1 0 0 0	24	0 + ((15-0)/31)*24 = 11.6
1 1 1 0 0	28	0 + ((15-0)/31)*28 = 13.6
1 1 1 1 1	31	0 + ((15-0)/31)*31 = 15.0

Note: *Variable min.value = 0, and variable max.value = 15

b) Real Value Coding Setup

To initialize the population in real value coding, flows for each canal in each chromosome of the population are generated randomly. For the problem investigate here, canal flows for the initial population were generated in the range of 0-15 m^3/s. Precision is not an issue.

4.4.2 Evaluation of Results

The criterias for the binary and real-value GAs are shown in Table 4.2. Results present as supply/demand ratios, and are given in Table 4.3 and Figure 4.5. Both representation schemes reached achieve near equity in supply.

Table 4.2. Criteria for GAs on test network

Representation scheme	**binary**	**real-value**
Population size	100	100
Avg.execution time in 500 generations (sec)	5.44	2.37
Number of genes	6	6
Number of alleles per gene	5	1
Length of chromosome	30	6
Probability of crossover	0.95	0.95
Probability of mutation	0.025	0.125
Number of mutations per chromosome	0.75	0.75
Type of selection	proportional	Proportional
Type of crossover	1-point	1-point
Type of mutation	conventional*	non-uniform

*conventional mutation is to mutate binary coding of 0 to be 1 or vice versa.

Table 4.3. Comparison of GA results in coding experiment

Node No.	**2**	**3**	**4**	**6**
Demand	4.000	5.000	5.000	3.000
Supply(binary)	3.548	3.871	4.839	2.903
Supply(real-value)	3.550	4.402	4.425	2.650
Supply/Demand ratio required	**0.882**	**0.882**	**0.882**	**0.882**
Supply/Demand ratio (binary)	0.887	0.774	0.968	0.968
Supply/Demand ratio (real-value)	0.887	0.880	0.885	0.883

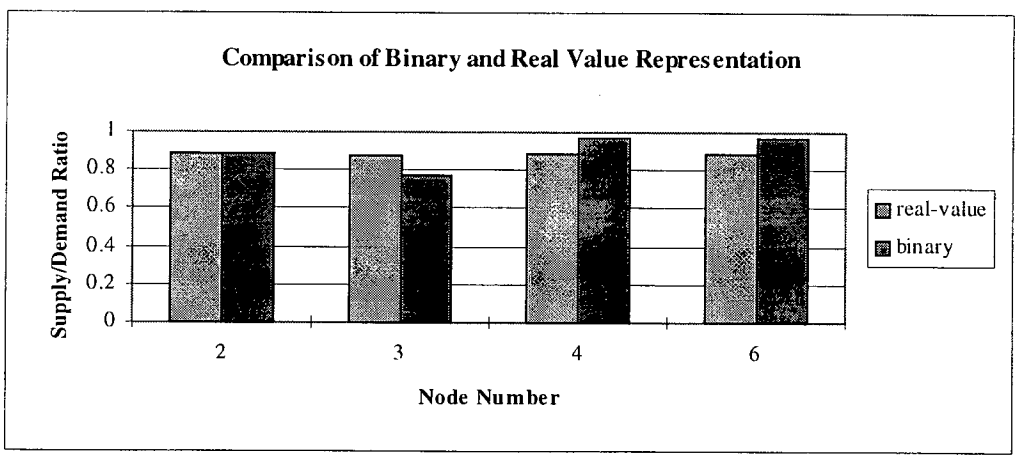

Figure 4.5 Supply/Demand ratio in the experiment of coding (a simple system)

An important difference between the binary and real-value codings is in the length of chromosomes. In binary coding there are five alleles per gene giving a chromosome length of 30. In real-value coding, there is only one allele per gene and the chromosome length is thus 6. A further disadvantage of binary coding is that coding and decoding slows down execution time. The average execution time for 500 generations with binary coding was 5 seconds while that with real-value coding was 2 seconds. It is clear that real-value coding progresses more quickly to a solution than does binary coding. Figure 4.6 shows the progress of the GAs towards solution with binary and real-value coding. Best fitness is the minimum fitness value (Z in equation 4.9) obtained as the run progress.

A further constraint on binary coding is the accuracy with which canal flows can be mapped to the binary bits, or alleles in a gene as indicated above. The linear mapping of values results in an accuracy of 0.484 m^3/s in each canal, and it is surprising given this that the results are as good as they are. To improve accuracy with binary coding, it would be necessary to include more alleles in each gene. This in turn increases string length and can result in slower execution and furthers problems in maintaining good solutions. The results achieved on this preliminary test demonstrate that GAs are capable of producing acceptable results, and that real-value coding is superior to binary coding.

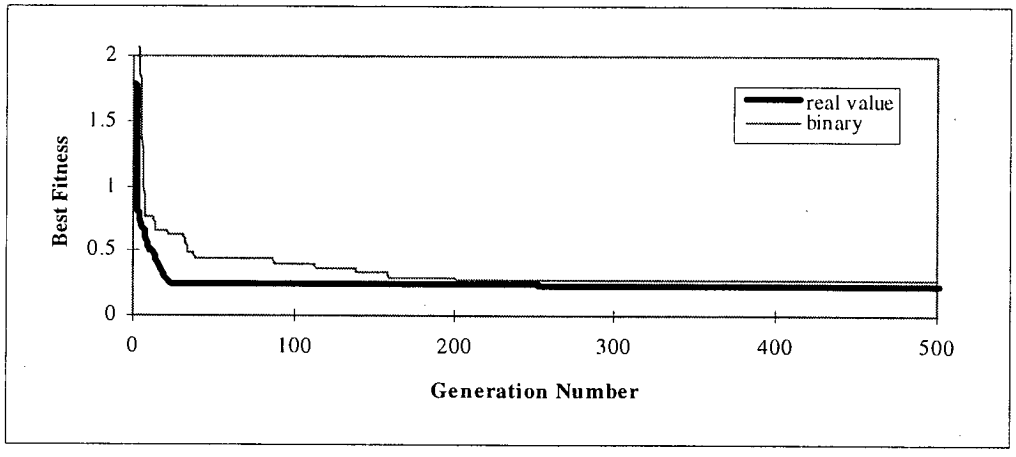

Figure 4.6 *Comparison of solution progress with binary and real value representation (a simple network)*

4.4.3 GA Performance with Strings Comprising Irrigation Supplies and Flows

A GA set up with supplies and flows represented in a chromosome requires a longer string length, but has the same constraints as a GA set up with flows alone. For the simple

network shown in Figure 4.3, the string comprises 12 genes instead of 6, as there are 6 irrigation supplies. A GA was set up using real-value coding, and results compared with those obtained earlier with a representation consisting of canal flows only.

The criteria for the alternative GA formulations are given in Table 4.4. Comparison of results for alternative formulations are presented in Table 4.5 and Figure 4.7. Clearly both approaches produce acceptable results, although a better results are produced by the formulation based on flows alone. The GA set up with flows and supplies took longer to reach an acceptable solution than the GA operating on flows alone. The execution time of GA operating on flows was 2.3 seconds for 500 generations, while that for the GA set up with flows and supplies was 2.6 seconds for 500 generations. Progression to solution is shown in Figure 4.8. It is clearly desirable to keep string length as short as possible.

The nodal balance constraint for the GA with both the flows and the supplies in a chromosome required an increased penalty factor to 5 greater than in the flows formation to ensure that constraints were satisfied. The reason for this is that random modification to two variables in the same constraint results in a greater chance of good solution being disturbed.

Table 4.4 Criteria for alternative GA formulations

Formulation of GA	**flows&supplies**	**flows**
Population size	100	100
Avg.execution time in 500 generations(sec)	2.64	2.37
Number of genes	10	6
Number of allele per gene	1	1
Length of chromosome	10	6
Probability of crossover	0.95	0.95
Probability of mutation	0.075	0.125
Number of mutation per chromosome	0.75	0.75
Type of selection	proportional	proportional
Type of crossover	1-point	1-point
Type of mutation	non-uniform	non-uniform

Table 4.5 Comparison of results for alternative formulations

Node No.	**2**	**3**	**4**	**6**
Demand	4.000	5.000	5.000	3.000
Supply (flows&supplies)	3.330	4.330	4.850	2.550
Supply (flows)	3.550	4.402	4.425	2.650
Supply/Demand ratio required	**0.882**	**0.882**	**0.882**	**0.882**
Supply/Demand ratio (flows&supplies)	0.832	0.866	0.970	0.850
Supply/Demand ratio (flows)	0.887	0.880	0.885	0.883

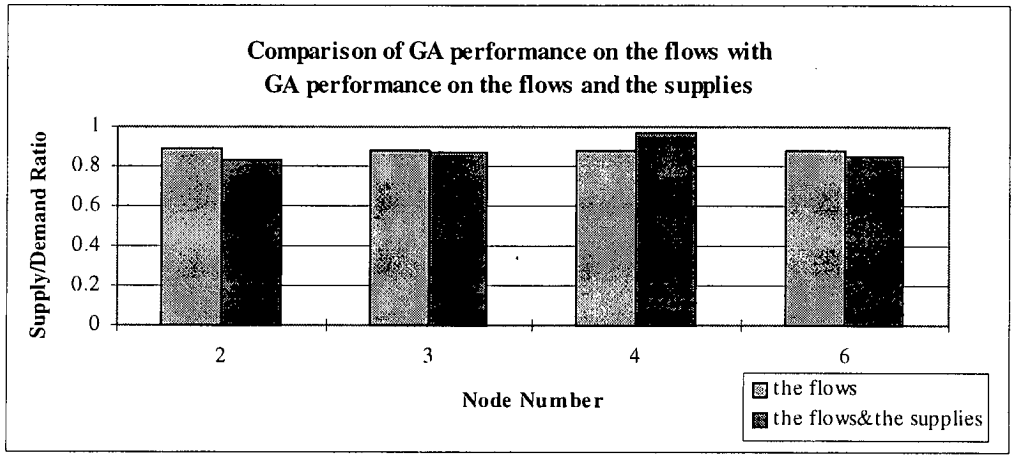

Figure 4.7 Supply/Demand ratio in the experiment of GA on the flows and the supplies (a simple network)

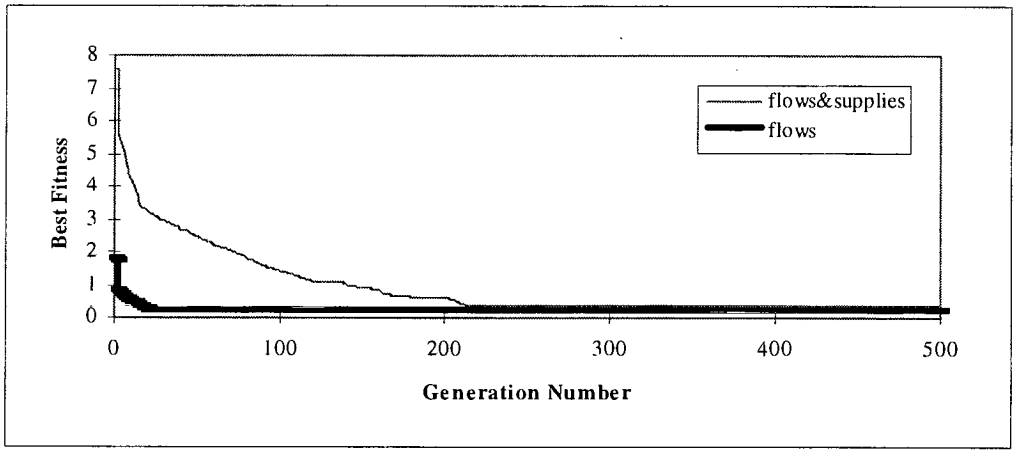

Figure 4.8 Solution progress with alternative formulations (a simple network)

4.4.4 Sensitivity Analyses on the Test Network

The preceding sections have demonstrated that real-value coding is superior to binary coding, and that the problem is best formulated using canal flows only. On the basis of this preferred set up, the sensitivity of the GA to crossover probability, mutation probability and population size has been investigated. The sensitivity to mutation probability is shown in Figure 4.9. The real-value coding is apparently insensitive to mutation probability, although best results are obtained with about 0.75 mutations per chromosome.

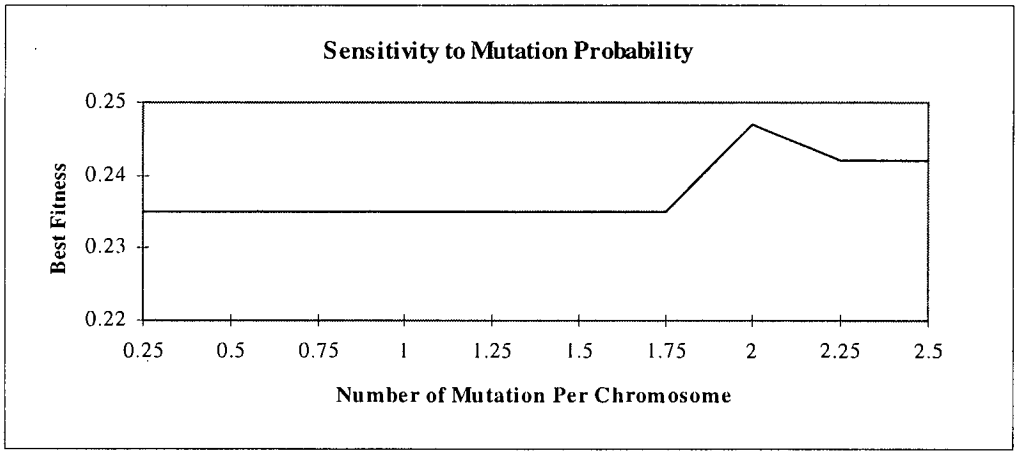

Figure 4.9 Sensitivity to mutation probability (a simple network)

This number of mutation per chromosome was used to test the sensitivity of performance to the probability of crossover. The sensitivity to crossover probability is shown in Figure 4.10. It would appear that the best results are achieved with a crossover probability of 0.95, although acceptable results are obtained over a fairly wide range. From the literature reviewed it has been suggested that the appropriate crossover probability is in the range of 0.5 to 1 (Goldberg 1989).

The sensitivity to population size is shown in Figure 4.11. The acceptable results are obtained within a wide range of population sizes. It would appear that the best results are achieved for the test network with a population size of 100 with crossover probability of 0.95 and the number of mutations per chromosome of 0.75. With too small a population there will be insufficient diversity, but with too large a population an increased number of generations will be required for the best individuals to emerge. In the runs reported here, the number of generations was fixed.

The sensitivity analysis indicates that the GA approach is robust and not significantly affected by parameter choices.

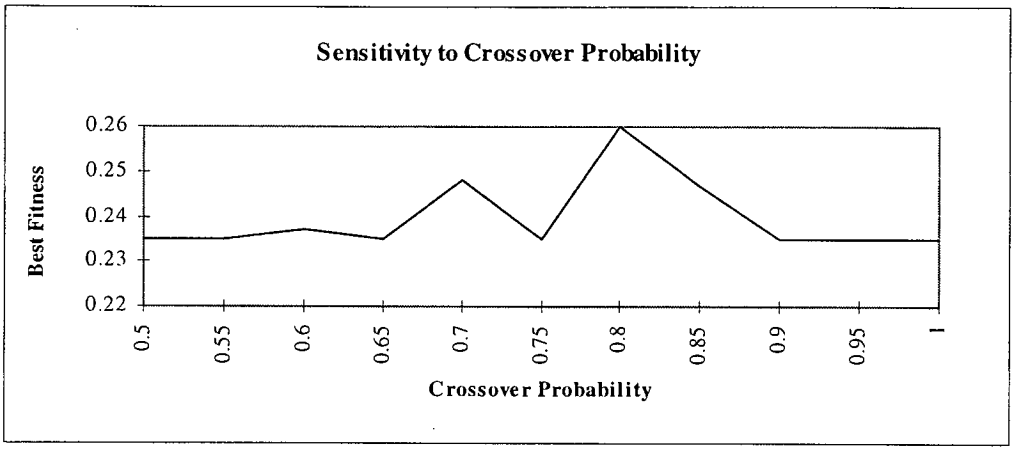

Figure 4.10 Sensitivity to crossover probability (a simple network)

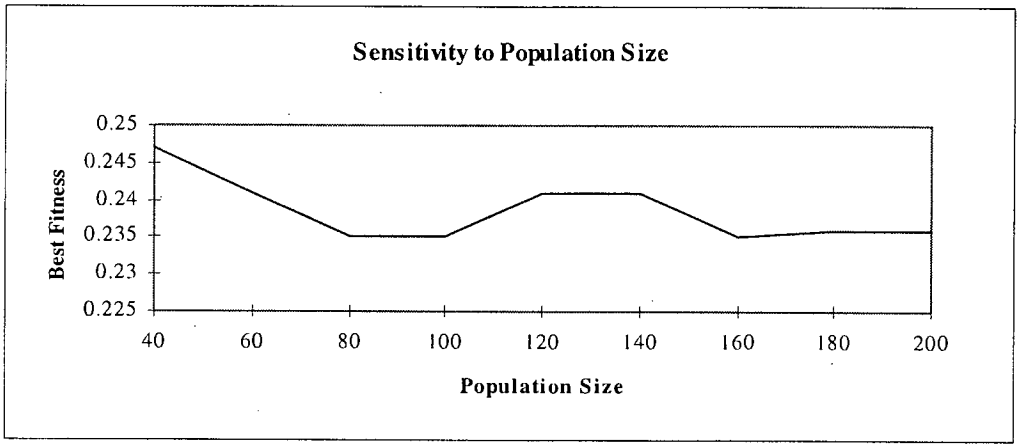

Figure 4.11 Sensitivity to population size (a simple network)

4.4.5 Operator Evaluation

An evaluation has been made of the alternative selection schemes (proportional and tournament selection), mutation schemes (conventional, modified mutation and non-uniform mutation) and crossover schemes (uniform crossover, one-point and two-point crossover). The different selection, mutation and crossover schemes are discussed in Chapter 3. The evaluation has been carried out using the GA formulated with flows alone, and real value representation. Results are shown in Table 4.6. It is observed that acceptable results could be found with all selection and crossover schemes, but that non-uniform mutation performs better than either conventional or modified uniform mutation. The advantage of non-uniform

mutation is that good solutions are not significantly disturbed by the mutation process (Michalewicz 1992).

Table 4.6 Evaluation of GA operators

Selection schemes	Best fitness
Proportional	0.235
Tournament	0.235
Crossover schemes	**Best fitness**
Uniform	0.235
One-point	0.235
Two-point	0.235
Mutation schemes	**Best fitness**
Conventional	0.357
Modified uniform	0.238
Non-uniform	0.235

Note: Non-uniform mutation is used with all crossover and selection schemes, and one-point crossover is used with all mutation and selection schemes.

4.5 Testing on a More Complex Network

In order to explore the potential of GAs more fully, the more complex network shown in the Figure 4.12 was devised. The network comprises 10 nodes and 12 reaches. Eight of the nodes are considered to be scheme nodes and have irrigation demands. The irrigation demands and inflows assumed are shown in Table 4.7. The objective function and constraints were as outlined in equations 4.2 to 4.5, and the exact same computer codes were used as for the simple network discussed in section 4.4.

Table 4.7 The demands and inflows for a more complex network

Node number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**		
Demand (m^3/s)	1.1	0.9	5.8	0	3.5	2.5	1.2	1.0	1.0	0.0		
Inflow (m^3/s)	14.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		
Reach number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Max.capacity (m^3/s)	3	15	8	3	8	2.5	5	2	3	4	2	2

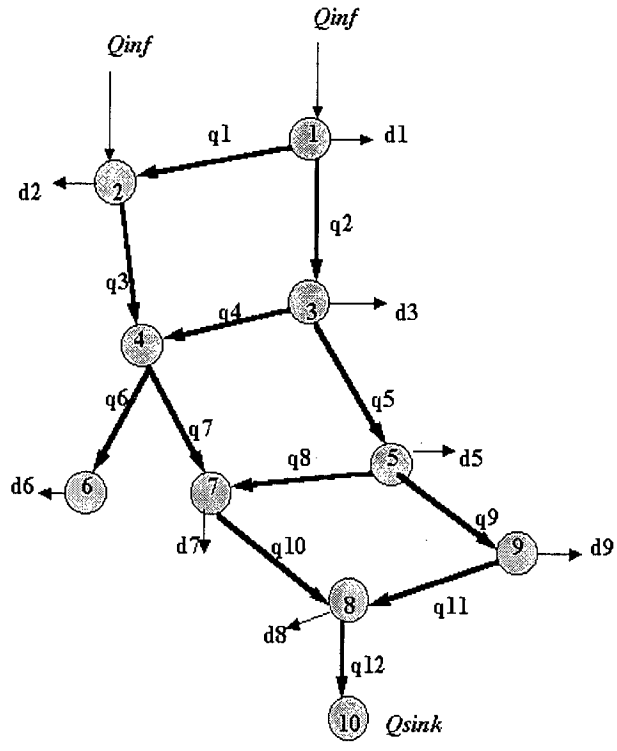

Figure 4.12 A test complex canal system

4.5.1 Evaluation of Representation Schemes

As with the simple network, a comparison has been made between the performance of GAs set up with binary and with real value representation. For each representation scheme, a population size of 100 was adopted with 0.75 mutations per chromosome, and a crossover probability of 0.95. The GAs were run for a maximum 500 generations. The GA characteristics are summarized in Table 4.8.

Results are summarized in Table 4.9. Both representation schemes have achieved acceptable results, but clearly real value representation has resulted in better equity. The

GA set up with real value coding executed in less than half the time required for that set up with binary representation, indicating clearly the advantage of real value representation over binary representation. The progress of both GAs to a solution is shown in Figure 4.13. The GA with real value representation clearly approaches the solution much more quickly, and is capable of achieving higher accuracy. In the GA with binary representation, 7 alleles were used in each gene, and with the mapping adopted this resulted in an accuracy of 0.118 m^3/s in reach flows. The string length for the binary representation was 7 times longer than that for the real value representation, and this combined with coding and decoding for fitness evaluation results in the longer execution times. The longer string length also makes maintaining good solutions more difficult, and this combined with the accuracy of representation results in a poorer overall performance than with real value representation.

Table 4.8 GA characteristics for alternative of representation schemes on the more complex network

Formulation of GA	**binary**	**real-value**
Population size	100	100
Avg.execution time in 500 generations(sec)	19.69	8.13
Number of genes	12	12
Number of alleles per gene	7	1
Length of chromosome	84	12
Probability of crossover	0.95	0.95
Probability of mutation	0.0089	0.0625
Number of mutation per chromosome	0.75	0.75
Type of selection	proportional	proportional
Type of crossover	1-point	1-point
Type of mutation	conventional	non-uniform

Table 4.9 Comparison of GA results with alternative representation schemes on the more complex network

Node No.	**1**	**2**	**3**	**5**	**6**	**7**	**8**	**9**
Demand	1.100	0.900	5.800	3.500	2.500	1.200	1.000	1.000
Supply (binary)	0.890	0.843	4.756	3.047	1.850	1.165	0.913	0.843
Supply (real value)	1.043	0.843	5.473	3.294	2.361	1.141	0.942	0.934
Supply/Demand ratio required	**0.941**	**0.941**	**0.941**	**0.941**	**0.941**	**0.941**	**0.941**	**0.941**
Supply/Demand ratio (binary)	0.809	0.936	0.820	0.871	0.740	0.971	0.913	0.843
Supply/Demand ratio (real value)	0.948	0.936	0.944	0.941	0.944	0.951	0.942	0.934

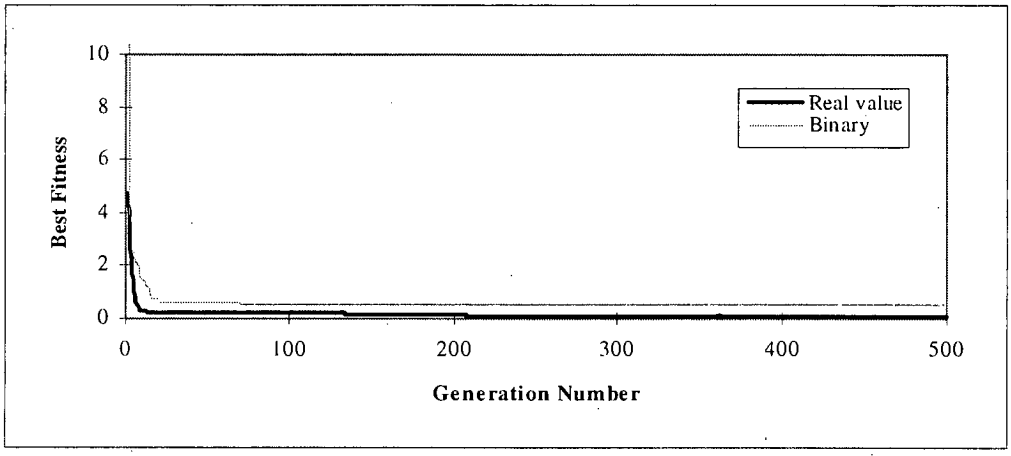

Figure 4.13 Solution progress with binary and real value representation on the more complex network

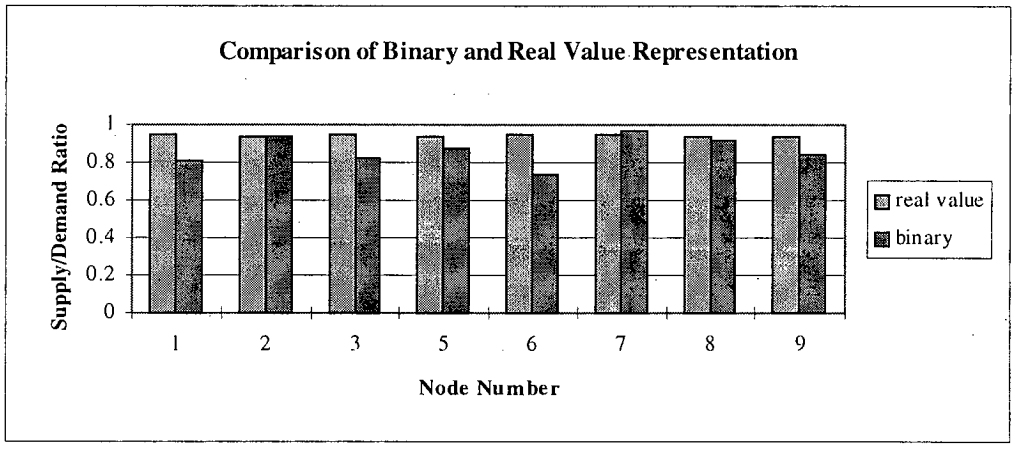

Figure 4.14 Supply/demand ratios with alternative representation schemes on the more complex network

4.5.2 Assessment of Alternative Formulations

As indicated earlier, there are two fundamental ways in which the GA can be set up. The first is with chromosomes comprised only of canal flows, and the second with chromosomes comprised of canal flows and irrigation supplies. When the GA is set up in terms of canal flows only, irrigation supplies are calculated from the nodal water balance. When the GA is set up in terms of canal flows and irrigation supplies, the chromosomes are longer, but the decision variables for the fitness function are part of the chromosomes and secondary evaluations are not required.

The GA characteristics for the alternative formulations on the more complex network are summarized in Table 4.10. Real value representation has been used in both formulations, and the only significant difference is therefore the number of genes in each formulation.

The GA based only on reach flows reached an acceptable fitness more quickly than the GA formulated on both canal flows and irrigation supplies. Figure 4.15 shows the manner in which fitness improves with generation for each of the approaches. This is as might be expected since the string length for the formulation involving flows and supplies is almost twice that of the formulation involving only canal flows. Consequently it takes longer to reach a solution. The additional water balance calculations required in the formulation based on canal flows do not add any significant overhead in execution time. The other problem with chromosomes formulated with both canal flows and supplies is that changes may be made through genetic operators to dependent variables, potentially upsetting good solutions.

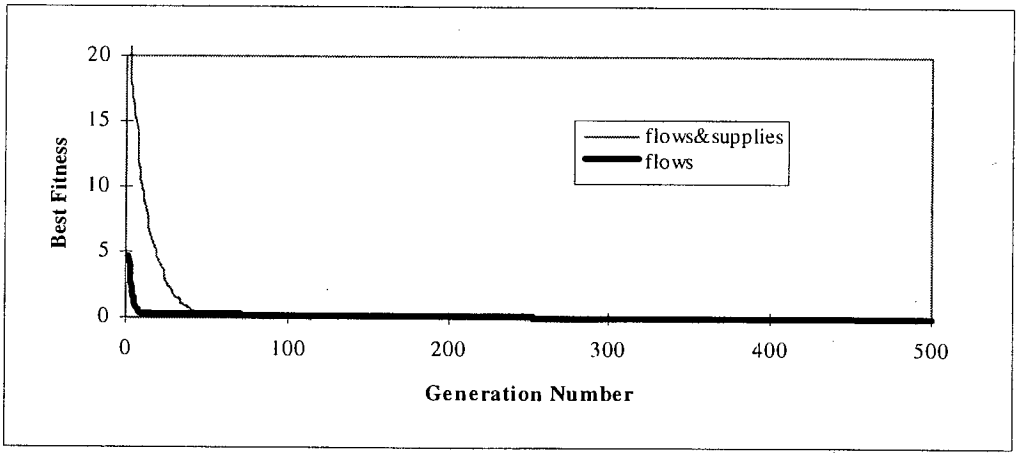

Figure 4.15 Solution progress with alternative formulations (complex network)

Table 4.10 GA characteristics with alternative formulations on the more complex network

GA formulation with	**flows&supplies**	**flows**
Population size	100	100
Avg. exe. time in 500 generations (sec)	10.37	8.13
Number of genes	20	12
Number of allele per gene	1	1
Length of chromosome	20	12
Probability of Crossover	0.95	0.95
Probability of Mutation	0.0375	0.0625
Number of mutation per chromosome	0.75	0.75
Type of selection	proportional	proportional
Type of crossover	1-point	1-point
Type of mutation	non-uniform	non-uniform

Table 4.11 and Figure 4.16 summarize the supply/demand ratios achieved with each formulation. The results achieved with both formulations are very good. Interestingly the formulation based on canal flows and irrigation supplies appears to work better on the more complex network than it did on the simple network discussed in section 4.4. There is, however, minor constraint violation on the more complex network, even after 500 generations. In the formulation based on canal flows and irrigation supplies, the total computed supply was 16.209 m^3/s, while inflows to the system were only 16.0 m^3/s. The formulation based on canal flows resulted in a small over supply of 0.031 m^3/s. With increasing string length, satisfying all system constraints is problematical, and this is a potential limitation of the GA approach to the water allocation problem. The constraint that appears to be most difficult to satisfy is nodal balance constraint. The formulation base on the flows and the supplies required performed best using nodal balance penalty function in quadratic form with penalty factor as shown in equation 4.11. Sensitivity to penalty factors of the formulation base on the flows and the supplies required an increased penalty factor up to 14, compared to a penalty factor 6 for the formulation base on only the flows.

$$P_2 = \sum_{i=1}^{n} \frac{(Qinf_i + \sum_{j=1}^{n} Q_{ij} - x_i - Qsink_i)^2}{Qinf_i + \sum_{j=1}^{r} Q_{ij(in)}} \qquad (4.11)$$

Table 4.11 Comparison of GA results with alternative formulations on the more complex network

Node No.	**1**	**2**	**3**	**5**	**6**	**7**	**8**	**9**
Demand	1.100	0.900	5.800	3.500	2.500	1.200	1.000	1.000
Supply (flows&supplies)	1.044	0.863	5.451	3.359	2.387	1.144	0.981	0.980
Supply (flows)	1.043	0.843	5.473	3.294	2.361	1.141	0.942	0.934
Supply/Demand ratio required	**0.941**	**0.941**	**0.941**	**0.941**	**0.941**	**0.941**	**0.941**	**0.941**
Supply/Demand ratio (flows&supplies)	0.949	0.958	0.940	0.960	0.955	0.953	0.981	0.980
Supply/Demand ratio (flows)	0.948	0.936	0.944	0.941	0.944	0.951	0.942	0.934

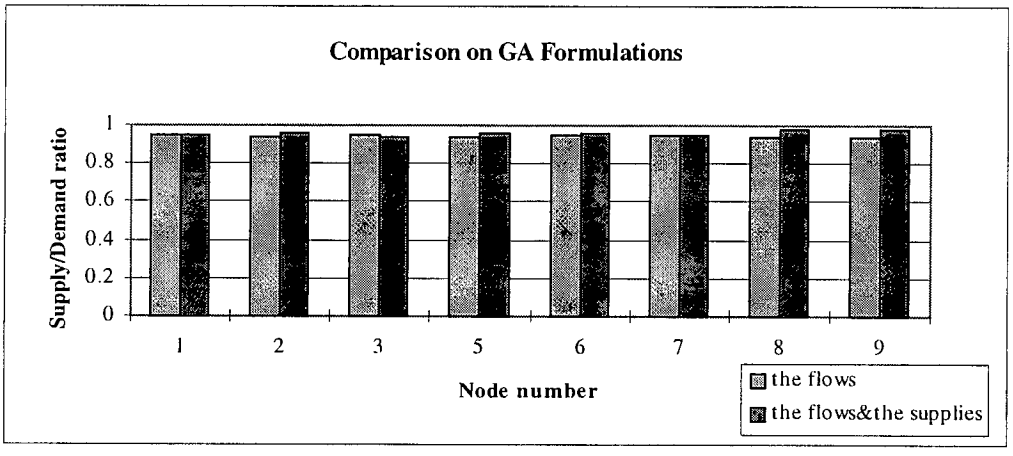

Figure 4.16 Supply/Demand ratio in the experiment of GA on the flows and the supplies (complex network)

4.5.3 Sensitivity Analyses on the More Complex Network

Sensitivity analyses have been carried out in the same manner as described for the simple network in section 4.4. The sensitivity analyses has been restricted to the GA formulated with canal flows only and real value representation. From the preceding analyses this is clearly the most robust set up.

Sensitivity to the mutation probability is presented in terms of the number of mutations per chromosome in Figure 4.17. Clearly there is little sensitivity to mutation probability in the range of 0.3 to 1 mutations per chromosome. The best fitness was obtained the same values when using 0.35, 0.5 and 0.95 mutations per chromosome. On the simpler network, a mutation probability of 0.75 mutations per chromosome was found to be most appropriate. From the results presented in Figure 4.18, this would seem that this gives acceptable results and is in the insensitive zone. Research on GA applications to reservoir systems by Wardlaw and Sharif (1999) also found that the best results would be achieved with about 1 mutations per chromosome with real value coding.

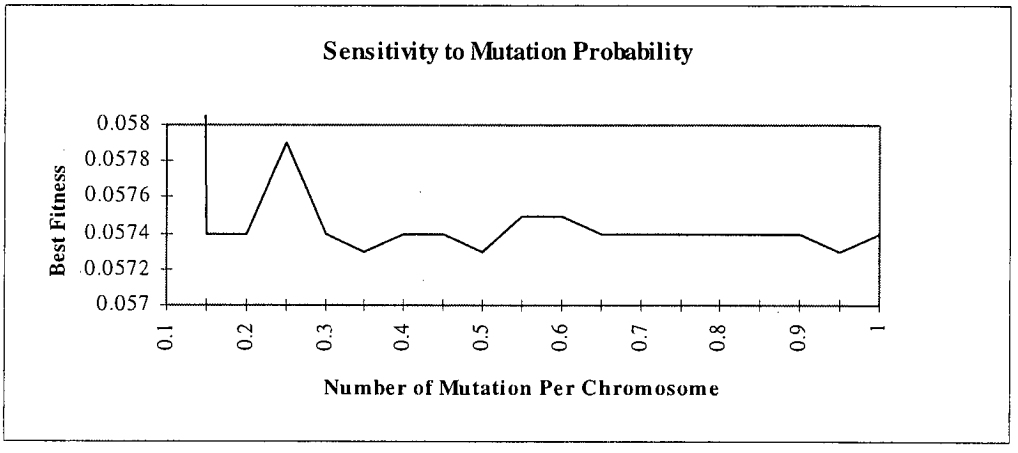

Figure 4.17 Sensitivity to mutation probability

Sensitivity to crossover probability is shown in Figure 4.18. With crossover probabilities in the range of 0.5 to 1.0, there is little sensitivity, and acceptable results are achieved. Work on the simple network indicated that a crossover probability of 0.95 would produce the best results. On the more complex network this is also the case, although similar results are achieved with values of 0.5, 0.7 and 0.9. Any crossover probability with the range of 0.5 to 1.0 produces good results. Wardlaw and Sharif (1999) found that on reservoir systems, there was similar insensitivity, and that a vale of 0.7 was most appropriate.

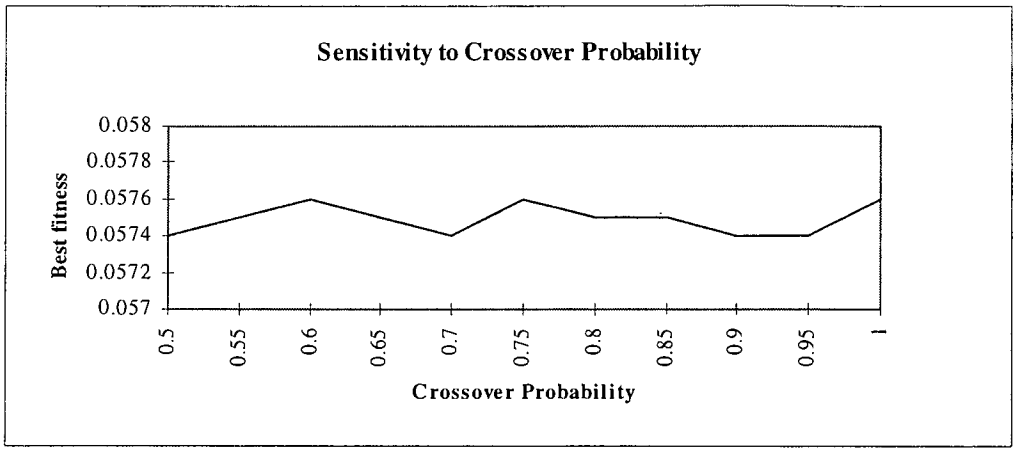

Figure 4.18 Sensitivity to crossover probability

Sensitivity to population size was analysed using a crossover probability of 0.95, and 0.75 mutations per chromosome. The results are shown in Figure 4.19. There is some improvement in fitness with increased population size, but this is not very significant. The supply / demand ratios with populations of 100 and 200 are summarised in Table 4.12. There is no significant improvement with a population size of 200, which results in increased execution times to 5 seconds in 500 generations. There was no improvement found on constraint violation when using population 200, compared with population size 100.

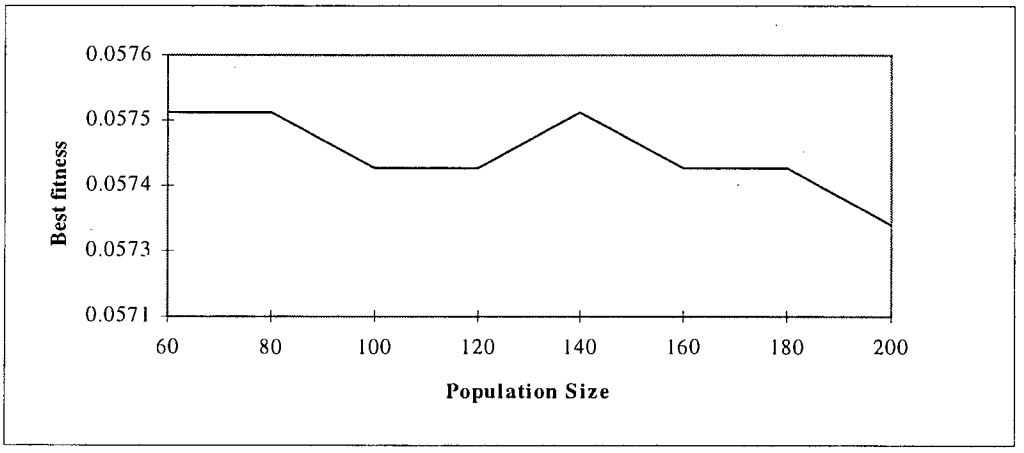

Figure 4.19 Sensitivity to population size

Table 4.12 Comparison on supply/demand ratio by population size

Node Number	**1**	**2**	**3**	**5**	**6**	**7**	**8**	**9**
Supply/Demand ratio required	0.941	0.941	0.941	0.941	0.941	0.941	0.941	0.941
with population size 100	0.948	0.936	0.944	0.941	0.944	0.951	0.942	0.934
with population size 200	0.945	0.934	0.944	0.943	0.927	0.946	0.957	0.928

4.5.4 Operator Evaluation

An evaluation has been made on alternative selection, mutation and crossover schemes. The evaluation has been based on real value representation and the formulation based on canal flows alone. Results are presented in Table 4.13. Tournament selection produced acceptable results but performed no better than proportional selection. This was not always be the case in other applications. Acceptable results can be obtained with all crossover schemes, as was the case with the simpler network discussed in section 4.4. The mutation scheme adopted does, however, have a significant influence on results. Non-uniform mutation performs much better than the other schemes. The more complex network with longer chromosomes is more sensitive to mutation than the simple network was, and non-uniform mutation helps preserve good solutions.

Table 4.13 Evaluation of GA operators

Selection schemes	Best fitness
Proportional	0.0575
Tournament	0.0596
Crossover schemes	Best fitness
Uniform	0.0575
One-point	0.0574
Two-point	0.0576
Mutation schemes	Best fitness
Conventional	0.5283
Modified uniform	0.1053
Non-uniform	0.0574

Note: Non-uniform mutation is used with all crossover and selection schemes, and one-point crossover is used with all mutation and selection schemes.

4.6 Discussion and Conclusions

Application of a GA to the water allocation problem in two test networks has permitted a number of useful observations and conclusions to be drawn:

i) A GA is capable of achieving acceptable results in terms of equitable water allocation on the networks tested, irrespective of the type of formulation or representation scheme used.

ii) A GA set up with real value representation executes more quickly and is more accurate than a GA set up with binary representation in application to the water allocation problem.

iii) A GA for the water allocation problem is best formulated in terms of canal flows alone, with irrigation supplies evaluated from nodal water balances.

iv) A GA applied to the water allocation problem with real value representation and formulated with canal flows is insensitive to crossover probability, or to the crossover scheme adopted. One-point crossover and a crossover probability of 0.95 produced good results on both test systems.

v) A GA applied to the water allocation problem with real value representation and formulated with canal flows is sensitive to the mutation scheme adopted. Non-uniform mutation performs significantly better than other schemes. The GA is, however, insensitive to mutation probability in the range of 0.3 to 1.0 mutations per chromosome. Good results were achieved in both test systems with a mutation probability of 0.75 mutations per chromosome.

vi) Constraint violation in the water allocation problem appears to be a potential limitation with larger more complex networks.

vii) A restriction on the range within which decision variables are initially randomly generated leads to quicker convergence.

5. APPLICATION OF GAs TO THE TUKAD AYUNG IRRIGATION SYSTEM

The results from GA application to the relatively simple systems discussed in Chapter 4 were promising, and led to the more demanding application to the Tukad Ayung Irrigation system in Bali. This system had been used by Wardlaw and Barnes (1996, 1997, 1998) in developing their quadratic programming approach to the water problem. It therefore provided a benchmark on which the GA performance could be measured directly against that of QP in a real irrigation system. A brief review of the results and findings of Wardlaw and Barnes was presented in Chapter 2. In this chapter, a more complete description of the irrigation system is given, along with the GA characteristics and formulations applied. Results are compared with those of Wardlaw and Barnes (1996) for the simplified modelling approach.

5.1 The Tukad Ayung Irrigation System

Bali is an island of the Republic of Indonesia. It is situated 9°south of the equator between Java island in the west and Lombok island in the east. Rivers flowing from the volcanic mountains in the center of the island provide water for irrigation (see Figure 5.1). Denpasar is the provincial capital of Bali. It is located in the southern part of the island and within the Tukad Ayung Irrigation System. The Tukad Ayung is the largest river in Bali and current supports over 14,500 ha of irrigated agriculture. This is the most fertile and productive agricultural land on the island. Rice is the principle crop, and over most of the area two rice crops and one dry foot crop are grown annually, under four basic cropping patterns. Denpasar is also the center of the booming tourist industry in Bali.

Studies of Tukad Ayung were undertaken by Sir M Macdonald and Partners (1988) as part of the Bali Water Resources Study. That study was concerned primarily with planning the means of meeting future potable water demands throughout the island, but with particular focus on Denpasar and the surrounding tourist areas. Establishing existing and likely future irrigation water use was an important part of the study, and to assist in this a mathematical model was developed for the Ayung catchment. Upstream of Buanga, a hydrological model

was used to extend stream flow records. Downstream of Buanga, a more complex irrigation system simulation model was developed to permit the impact of increasing potable water abstraction on agricultural production to be assessed. These models are described by Wardlaw and Wells (1996). The irrigation system is complex. Schematics have been shown in Figures 2.3 and 2.4. Significant drainage re-use occurs. Some drainage contributes laterally to other irrigation schemes and some is returned directly canals. Figure 5.2 shows proportional splits in drainage returns.

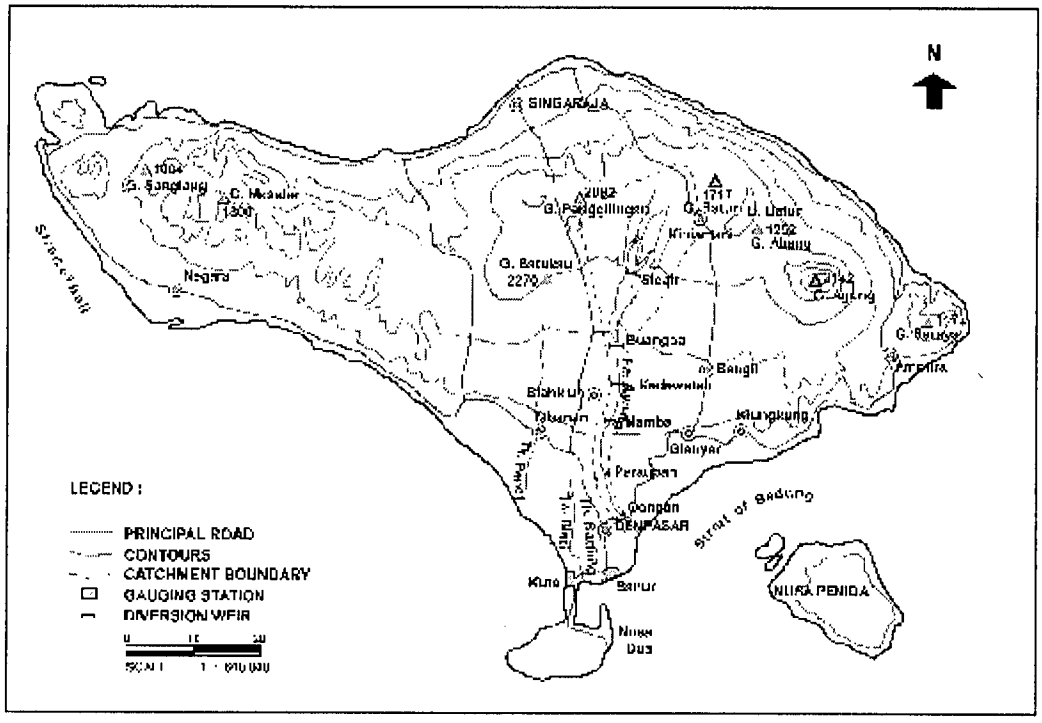

Figure 5.1 The Island of Bali (After Wardlaw and Barnes 1996)

5.2 Tukad Ayung Irrigation Network Definition

As part of their research, Wardlaw and Barnes (1996) developed a transparent matrix approach to irrigation network definition. The same approach has been adopted in setting up the GAs for the system, and the data files prepared by Wardlaw and Barnes could therefore be used directly with the GAs. The network definition file is included in Appendix B. The network comprises 69 reaches and 56 nodes. Of the 56 nodes, 10 are inflow nodes, 13 are scheme nodes, and a further 10 are sink nodes. Sink nodes represent system outflows. A summary of the nodal types which may be related to Figure 5.2 is given in Table 5.1.

Table 5.1 Ayung irrigation system summary of node information

Node Type	Nodes Numbers
Inflow	3 10 14 21 43 40 54 61 69 85
Scheme	106 6 122 22 127 27 45 50 6 71 77 81 88
Sink	24 36 66 72 78 79 83 89 92 99

Wardlaw and Barnes (1996) ran their water allocation model on a historic 34 year simulation period, using bimonthly time steps. Through this it was possible to evaluate model performance under a wide range of conditions. The 34 year simulation period had been set up by Sir M MacDonald and Partners (1989) to permit evaluation of water supply and irrigation production reliabilities. Inflows were defined through hydrological simulation, and the files used in model input are included in Appendix B. Irrigation demands used were design irrigation demands, based on effective rainfalls at 20% non-exceedance probability, and cropping patterns in existence at the time of the study. The irrigation demands file is also included in Appendix B.

5.3 GA Formulation

The GA was formulated using canal flows as the decision variables, with real value representation, uniform crossover, non-uniform mutation, proportional selection and incorporating the elitist strategy.

Initially the GA was formulated with chromosomes comprising flows in every canal of the network. Chromosomes thus comprised 69 genes. This formulation was referred to as GA1.

5.3.1 The Objective Function Used

The objective function, or evaluation function used in the GA was basically that of equation 4.8. It was found in preliminary applications, however, that constraint violation was difficult control as a result of the long strings used. Some hint of potential difficulties of this nature had arisen from the tests outlined in Chapter 4 on a more complex network. The indications there were also that the global water balance was not always being satisfied. A further constraint was therefore introduced to control the global water balance and avoid creep induced through errors in nodal balances. The following additional penalty factor was therefore introduced to equation 4.8:

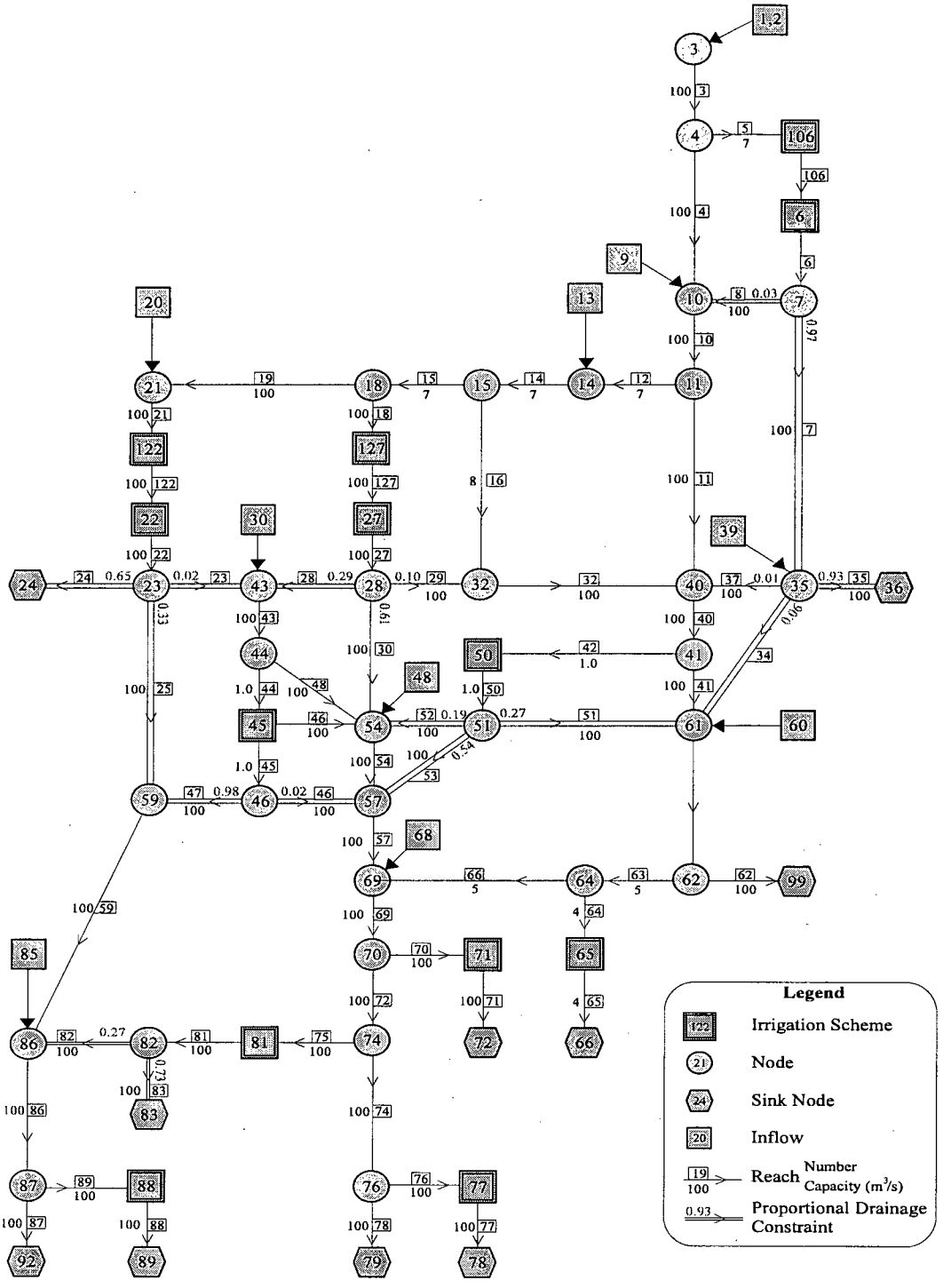

Figure 5.2 Nodal schematic network of Ayung irrigation system

(After Wardlaw and Barnes 1996)

$$P_4 = \frac{(\sum_{i=1}^{n} Qinf_i - \sum_{i=1}^{n} X_i - \sum_{i=1}^{n} Q\,sink_i)}{\sum_{i=1}^{n} Qinf_i} \qquad (5.1)$$

where,

$Qinf_i$	=	external inflow to node i (i.e. not from the canal system)
X_i	=	water supply to irrigation scheme *i*
$Qsink_i$	=	outflow to system sink
i, j	=	node reference
n	=	number of irrigation schemes
Q_{ij}	=	flow to node *j* from node *i*
P_4	=	penalty for the global water balance constraint

It was also found that with 56 nodes at which water balance constraints must be satisfied, the GA was particularly sensitive to the weighting attached to this. It was found that a quadratic form of the nodal water balance constraint was most appropriate:

$$P_2 = \sum_{i=1}^{n} \frac{|Qinf_i + \sum_{j=1}^{m} Q_{ij} - x_i - Qsink_i|^2}{Qinf_i + \sum_{j=1}^{r} Q_{ij(in)}} \qquad (5.2)$$

where,

$Q_{ij(in)}$	=	flow(s) into node *i*
m	=	number of reaches connected to node *i*
r	=	number of reaches flow into node *i*

The extended evaluation function used was therefore:

$$\text{Minimise } Z = \sum_{i=1}^{n} \frac{(d_i - x_i)^2}{d_i} + R_1 \cdot P_1 + R_2 \cdot P_2 + R_3 \cdot P_3 + R_4 \cdot P_4 \qquad (5.3)$$

Sensitivity analysis was carried out on the weightings for the penalty factors and is presented in heading 5.7

5.3.2 Convergence Criteria

The GA was set up to run on 34 years of bimonthly data, and the number of generations required was found to vary from time step to time step. The maximum number of generations was initially set at 500. However, if the minimum fitness is still improving after 500 generations, the GA was allowed to run in the multiple of 500 generations. The convergence criteria set were that the improvement in minimum fitness over a generation gap of 300 must be less than 0.0001.

5.3.3 GA1 Run Characteristics

The run characteristics for GA1 are summarised in Table 5.2. It was found that a low error tolerance was required (i.e. high penalty factors for constraint violation) as a result of the complexity of the system and long chromosomes. In time steps with no water stress, the nodal water balance constraints were satisfied to a precision of 0.008 m^3/s. The GA performed poorly on nodal balance constraints. In time steps with water stress, constraints were rarely satisfied to this precision. It was found that the nodal water balance error was typically in the range of 0.0 to 0.1 m^3/s after average of 800 to 1800 generations.

Table 5.2 Run characteristics for GA1 on the Tukad Ayung system

Items	**Results**
Population size	100
Avg. execution time per time step (minute)	1.31
Number of genes	51
Number of alleles per gene	1
Length of chromosome	51
Probability of crossover	0.95
Probability of mutation	0.0147
Number of mutation per chromosome	0.75
Type of selection	proportional
Type of crossover	uniform
Type of mutation	non-uniform

5.4 Comparison of GA1 and QP Results

Average irrigation deficits resulting with GA1 approach are compared with those achieved by Wardlaw and Barnes (1996) using QP in Figure 5.3. The QP approach clearly performs much better in terms of equity of supply, and has produced lower deficits. The average annual irrigation demand is 11.5 m^3/s. Using QP the average supply per time step was 10.27 m3/s while with GA1 the average supply was 9.68 m^3/s. Average outflow per time step with GA1 was 3.16 m^3/s while that with QP was 2.47 m^3/s. Figure 5.4 shows the comparison of average outflows with the GA1 and QP approaches. The GA1 formulation did not perform well in terms of equity or total supply. Execution time with the GA was also significantly slower than that of the QP approach. The average execution time per time step for GA1 was 36 times longer than that of the QP.

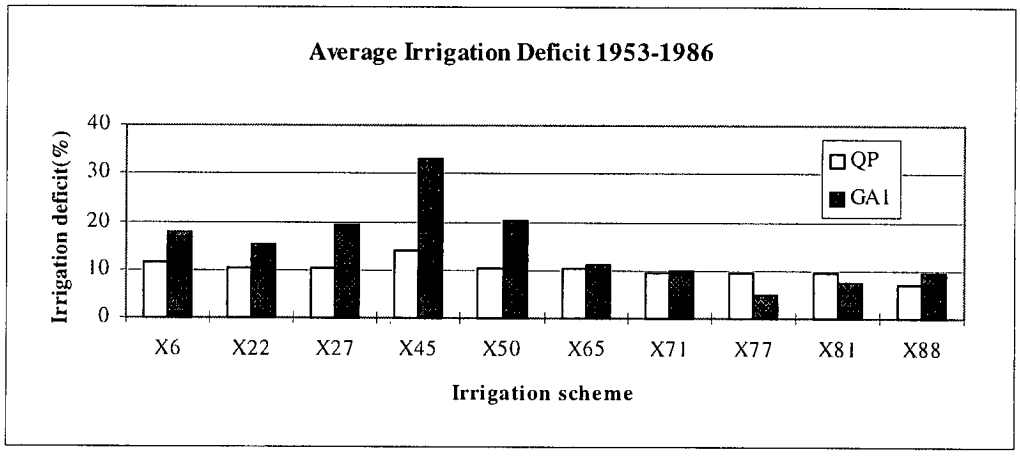

Figure 5.3 Comparison of average irrigation deficits (1953-1986)

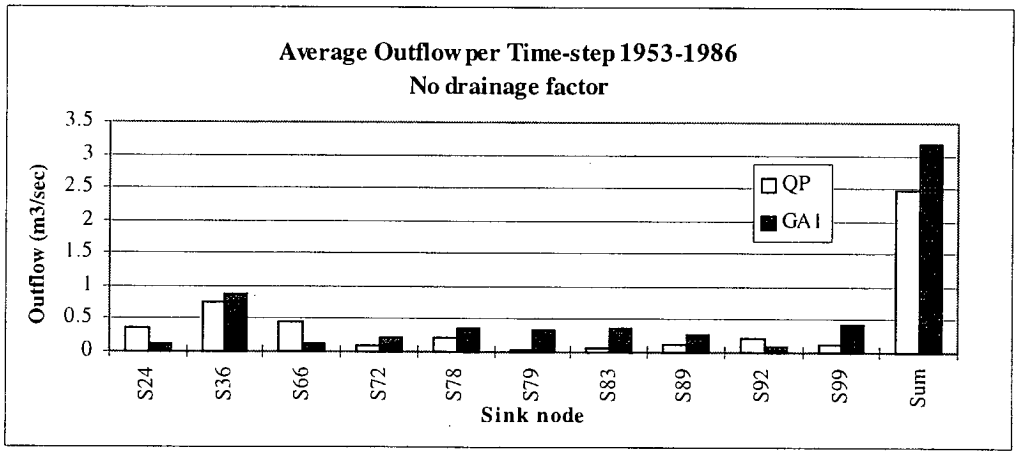

Figure 5.4 Comparison of average outflows (1953-1986)

5.5 Development of an Improved GA Formulation

It was considered that part of the reason for the poor results achieved with GA1 were related to difficulties in achieving nodal water balances, and the fact that at certain nodes in the network, where there were no irrigation demands, all flows in all canals meeting at these nodes were being treated as decision variables. At such nodes, flow in one reach can be computed from the nodal water balance and treated as a dependent variable, thus reducing the number of decision variables and gene length. In this revised formulation (GA2), the gene length was reduced from 51 to 25 and the potential for constraint violation reduced considerably also.

The transparent matrix approach to irrigation network definition of Wardlaw and Barnes (1996) has been extended. A reach index identifies those reaches in which flows are to be computed from the nodal balance. The extension of new network data file is shown in Appendix B.

The objective function used was exactly the same as that used with GA1. GA2 was set up with the same run characteristics as GA1 also - population size of 100, crossover probability of 0.95, and 0.75 mutations per chromosome. The run characteristics are summarised in Table 5.3.

The improved GA progresses to a solution more quickly than GA1 and achieves better fitness. Figure 5.5 shows how fitness improves by generation with GA1 and GA2. The best fitness after 1800 generations was 4.8 from GA2, and 5.6 from GA1. A comparison of average irrigation deficits with QP, GA1 and GA2 is presented in Figure 5.6. GA2 clearly performs significantly better than GA1, but still does not produce the same level of equity as the QP approach. A comparison of average irrigation supplies to each scheme is presented in Table 5.4. The average total supply with GA2 is 10.12 m^3/s, compared with 10.27 m^3/s with the QP. The average outflow per time step with GA2 was 2.62 m^3/s, compared with 3.16 m^3/s with the GA1.

Table 5.3 Criteria formulation for improved GA (GA2)

Items	Results
Population size	100
Avg. execution time per time step (minute)	1.21
Number of genes	25
Number of alleles per gene	1
Length of chromosome	25
Probability of crossover	0.95
Probability of mutation	0.03
Number of mutation per chromosome	0.75
Type of selection	proportional
Type of crossover	uniform
Type of mutation	non-uniform

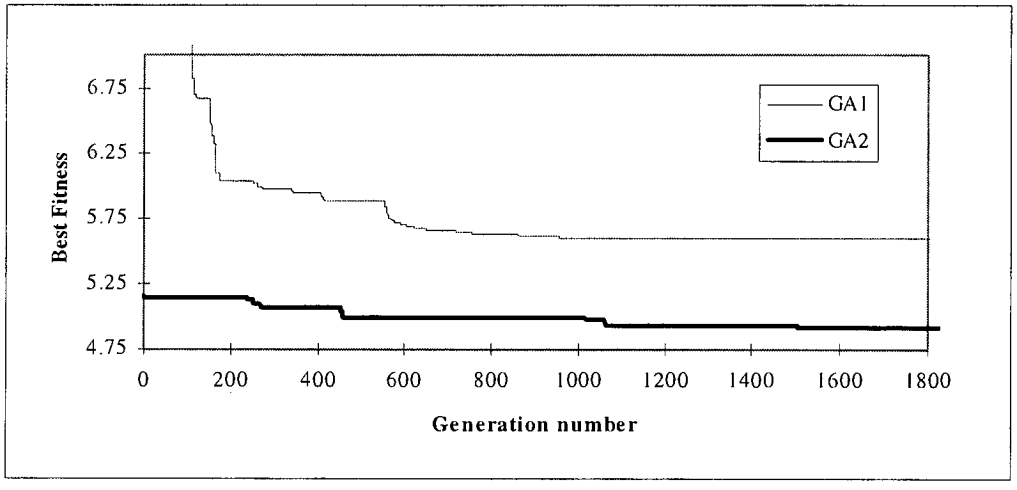

Figure 5.5. Solution progress with GA1 and GA2

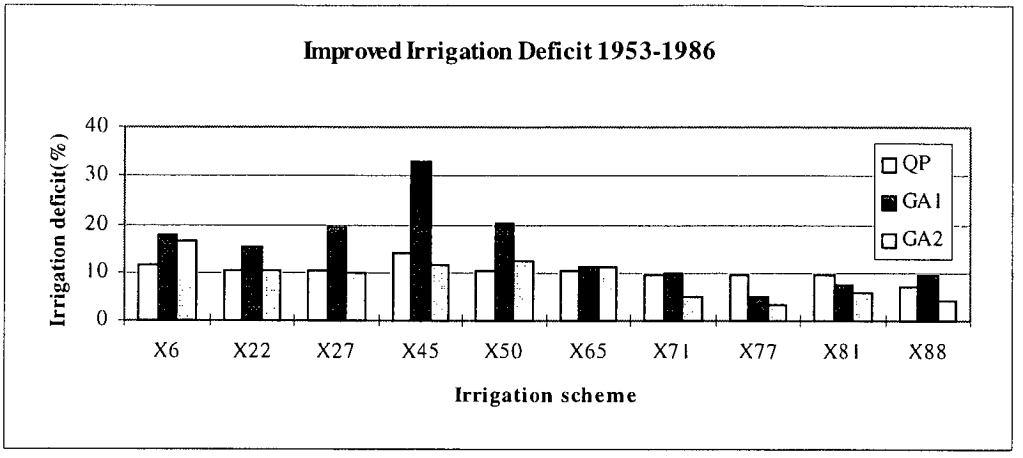

Figure 5.6. Irrigation deficits with GA2

Table 5.4 Comparison of average irrigation supplies

Scheme	Demand	Supply		
		QP	GA1	GA2
X6	3.91	3.46	3.22	3.27
X22	3.86	3.46	3.27	3.46
X27	0.89	0.80	0.72	0.80
X45	0.36	0.31	0.24	0.32
X50	0.15	0.13	0.12	0.13
X65	1.20	1.08	1.07	1.06
X71	0.30	0.27	0.27	0.28
X77	0.34	0.31	0.32	0.33
X81	0.30	0.27	0.28	0.28
X88	0.20	0.19	0.18	0.20
Avg. per time step(m^3/s)	11.50	10.27	9.68	10.12

5.6 Comparison on Selection Operators

The application of a GA to the water allocation problem was initially achieved using proportional selection. Additional runs have been carried out to analyse how tournament selection will effect the results. The evaluation has been tested base on the generated GA2. The GA was set up to run in the 34 years from 1953 to 1986 using the same run characteristics and convergence criteria of GA2.

It was found that tournament selection was also able to achieve similar results to those achieved by proportional selection. With tournament selection the average irrigation supply per time step was 10.04 m^3/s, compared with 10.12 m^3/s with proportional selection. Figure 5.7 presents the comparison of the average irrigation supply per time step. Average outflow per time step with proportional selection was 2.62 m^3/s compared with 2.75 m^3/s with tournament selection.

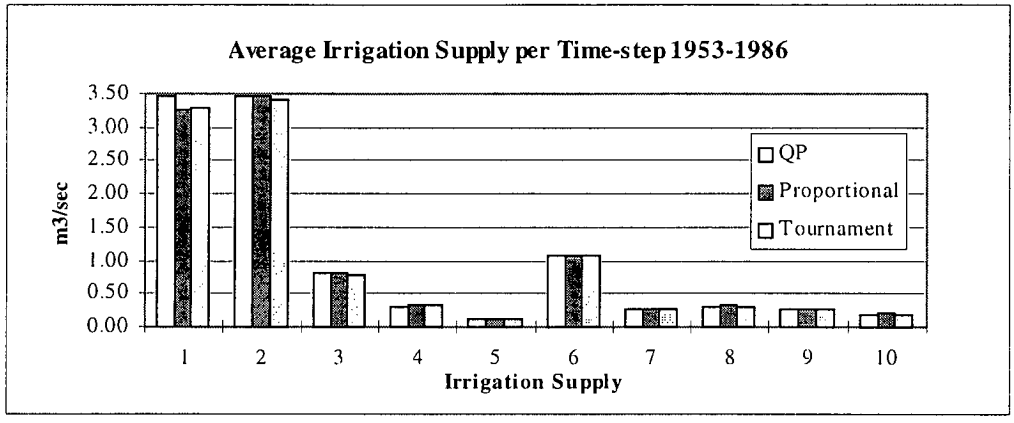

Figure 5.7 Comparison of selection operators on average irrigation supplies

5.7 Sensitivity Analyses

Sensitivity analysis has been carried out with GA2 on the Tukad Ayung network in order to assess if the findings of the sensitivity analysis on the test networks was valid on more complex networks. Time step 18 of 1986 was chosen for the analysis as this time step had significant water stress – total inflows were only 7.55 m^3/s while demand was 16.576 m^3/s. Sensitivity to mutation and crossover probabilities have been carried out with proportional selection. Results are shown in Figures 5.8 and 5.9 respectively. It was found that 0.75 mutations per chromosome produced good results, and in fact there is little sensitivity to mutation probability. Crossover probabilities in the range of 0.5 to 1 were considered. These also show little sensitivity. A crossover probability of 0.95 was considered to be appropriate for model runs.

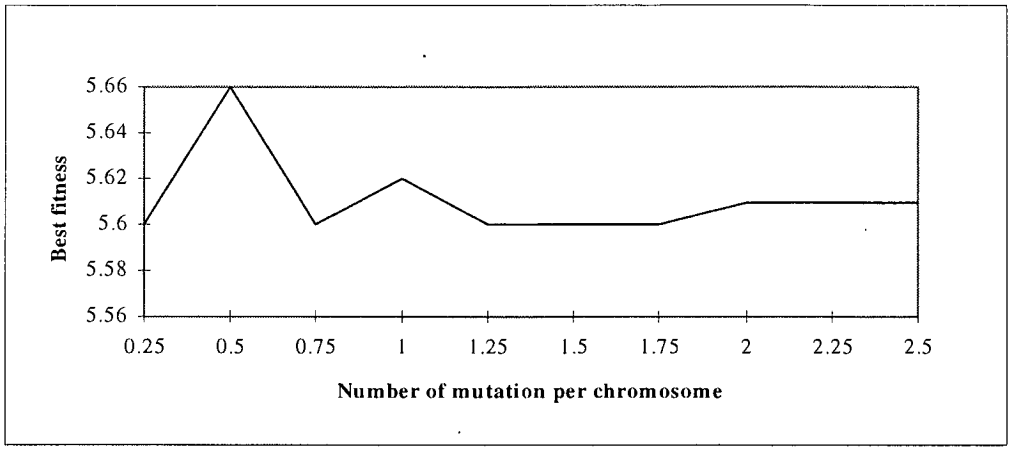

Figure 5.8 Sensitivity to mutation probability

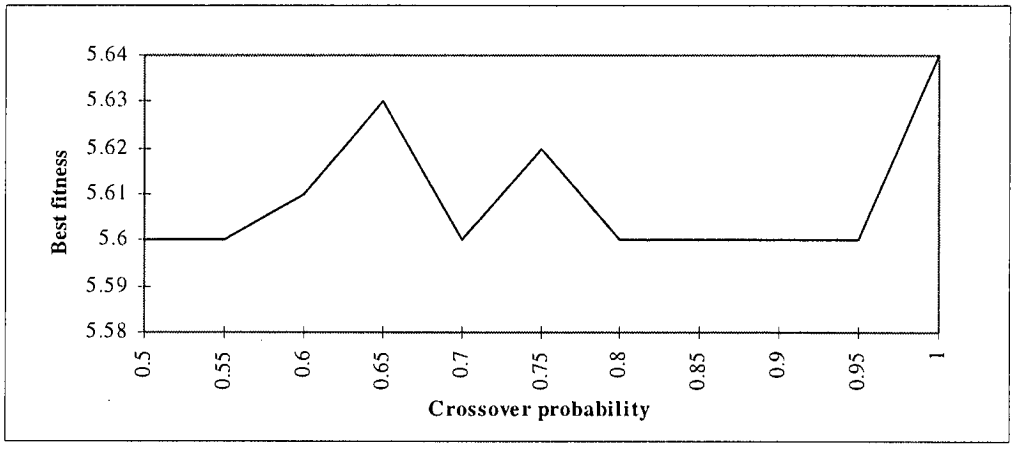

Figure 5.9 Sensitivity to crossover probability

Sensitivity to population size is shown in Figure 5.13. Fitness improves as the population increases to 100, but no improvement was found after that, and a population size of 100 was considered to be appropriate for model runs. Population size significantly affects execution time, as can be seen from Table 5.5, and in application to multiple time step runs, as were used in Bali, execution times are of importance. It will also be noted from Table 5.5 that the GA2 execution time is almost 40 times longer than that of the QP approach.

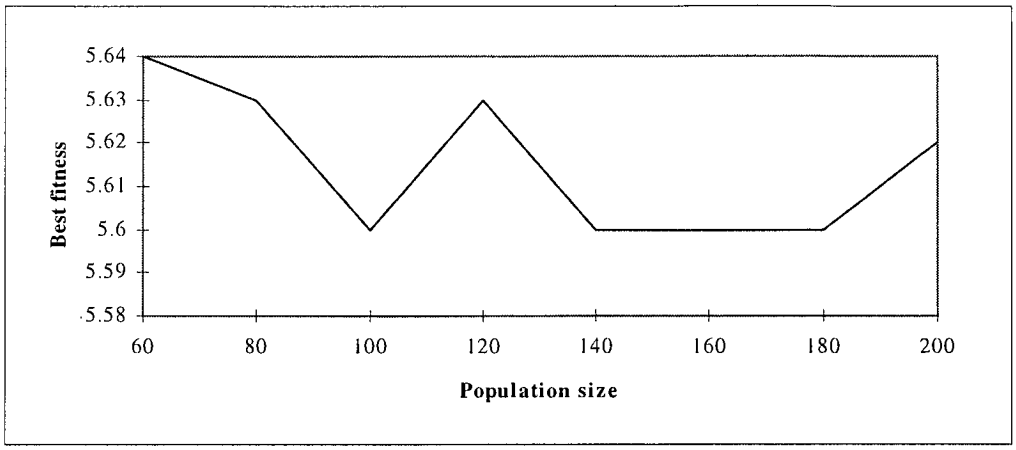

Figure 5.10 GA2 sensitivity to population size

Table 5.5 Influence of population size on execution times

	Population Size		
	100	200	300
Avg. GA2 exec.time per time step (sec)	78.6	176.4	268.4
Avg. QP exec. time per time step (sec)	2.2		

Sensitivity to penalty factor R2 (nodal balance constraint) has been investigated using the whole inflow data in 1986, 24 time steps. Figure 5.11 indicates that the best fitness achieved with R2 = 1 with equation 5.2.

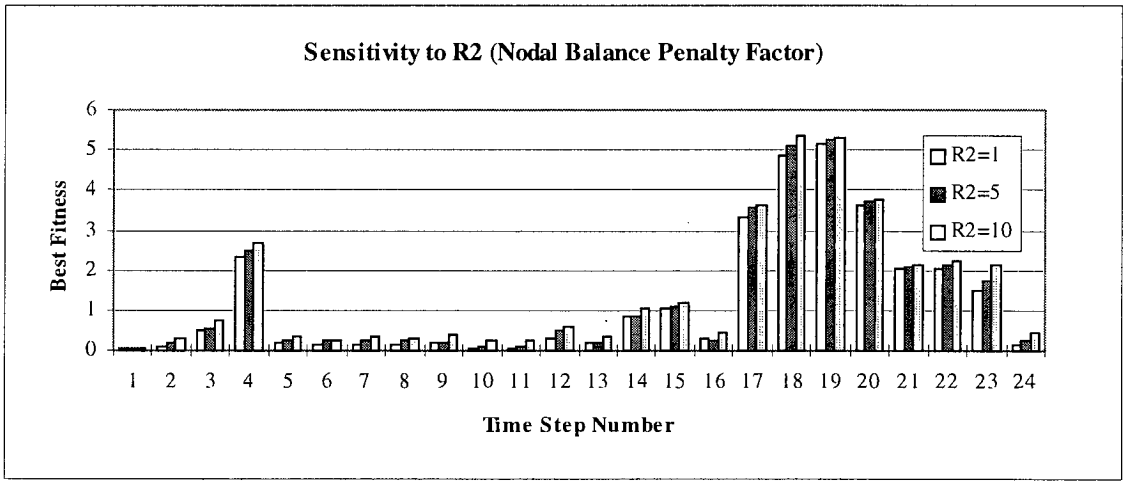

Figure 5.11 Sensitivity to penalty factor R2, nodal balance constraint.

5.8 Discussion and Conclusions

It has been demonstrated that with the improved GA formulation (GA2), solutions to the water allocation problem can be achieved that are similar to those achieved by QP. The GA should be set up with canal flows as the decision variables. The number of decision variables can be reduced by making use of continuity at junctions with no irrigation demands to permit one canal flow to become a dependent variable.

It is concluded that a real-value representation, proportional selection together with the elitist strategy, uniform crossover and non-uniform mutation will produce the best result. A crossover probability of 0.95 and 0.75 mutations per chromosome are suitable for the water allocation problem. The approach has been shown to be robust and not particularly sensitive to crossover or mutation probabilities. It is sensitive to population size, however, and too small a population restricts diversity. Too large a population will not improve fitness but will increase execution times.

The GA is clearly sensitive to string length, and this will limit its application to larger problems. As string length increases, maintaining good solutions becomes more difficult. The closed nature of the water balance at supply nodes makes solution by GA difficult. The main advantage of the GA approach is that it can be set up with any form of objective function. However, for the water allocation problem, although the improved GA produced acceptable results, it offers no real advantage over the QP approach. Execution times for the QP approach were significantly shorter than those of the improved GA. It is thought that the GA does hold promise for application to water scheduling problems in the next chapter.

6. THE WATER SCHEDULING PROBLEM

6.1 Introduction

In the previous studies by Wardlaw and Barnes (1996, 1997, 1998), and in the research described in the earlier chapters of this thesis, it was appropriate to consider continuous water supplies to each irrigation scheme, and scheduling within a scheme was of no relevance to the optimisation approach. Consideration is given here to irrigation systems in which problems relate primarily to scheduling within rotational systems. It was considered that GAs might be better suited to this type of problem than they are to the water allocation problem.

6.2 Literature Review

Rotation of irrigation is common in most systems at some level, and optimisation techniques have been applied to water scheduling problems by a number of authors.

Rao et al. (1992) studied the problem of real time irrigation scheduling under water shortage conditions. They took soil moisture content and available water supplies as state variables characterizing the irrigation scheduling problem. The objective was to develop irrigation schedules that would maximise crop yields. The decisions were made in two stages, a design stage and a real time stage. At the design stage, irrigations were planned for weekly intervals using historical data of seasonal supply, probable weekly rainfall, average weekly potential evapotranspiration, and basic data on soils, crops and the irrigation system. A standard soil moisture deficit model was used to obtain weekly design irrigation requirements for the season. In real-time management the decisions for the subsequent weeks were revised at the end of each week after updating with the real time values of rainfall, evapotranspiration, and the availability of water supply. If actual rainfall, evapotranspiration and available water supply were significantly different from values input at the design stage, then the optimization model would be re-run for the subsequent weeks in the season. The model was demonstrated to be effective.

In an irrigation scheduling study carried out by Naadimuthu et al. (1999), the objective was to maximise the net revenue from corn ($/ha). The study was based on a dynamic crop response model for corn and an evaporation or soil moisture balance model. State variables were the potential yield on day *t* during the reproductive phase, the amount of available soil moisture during period *t*, stage number (days), and the sum of evaporation from the soil surface during day *t*. A heuristic dynamic optimisation method, which reduced the four state variables to a one variable problem was developed, and the problem solved by using the available soil moisture during period *t* as the state variable. Other variables were the price of corn, the potential yield on any day during the productive phase, the number of days from the day of emergence to maturity, the amount of irrigation during period *t*, and the total cost of irrigation. The constraints included the water balance equations through which the available moisture content was obtained at stage *t*. At any stage it was assumed that the soil moisture level would be brought to field capacity by irrigation in future days. The study found, as have many others, that the main difficulty of dynamic programming relates to the number of state variables. The approach developed by Naadimuthu et al. to reduce the number of state variables was effective. The accuracy of the heuristic dynamic optimisation algorithm depends on the number of iterations and the step size used. The heuristic algorithm was shown to be more efficient than a standard dynamic programming approach.

Sunantara and Ramirez (1997) studied optimal seasonal multicrop irrigation water allocation and optimal daily irrigation scheduling using a two-stage decomposition approach based on stochastic dynamic programming. The goal was to optimize the seasonal allocation of a limited amount of irrigation water as well as to optimize the seasonal allocation of limited acreage for two or more crops, and to determine the optimal daily irrigation scheduling policy for each crop, taking into account the dynamics of the soil moisture depletion process and the stochasticity of rainfall. In stage1 (seasonal water allocation) the optimization model was based on seasonal crop production functions using a single-crop stochastic dynamic programming irrigation scheduling model to determine the maximum expected value of benefits for a single crop as a function of seasonal water availability. This incorporated the physics of soil moisture depletion and the stochastic properties of precipitation. The optimal seasonal water allocation between several fields was made using deterministic dynamic programming. The objective was to maximize total benefits from all crops. In stage2 (intraseasonal water allocation) optimal intraseasonal irrigation scheduling was performed using a single-crop stochastic dynamic programming algorithm, conditioned on the optimal seasonal water allocation of stage one. The daily optimal irrigation scheduling functions

were obtained as a function of root-zone soil moisture content and the currently available irrigation water. The study showed a strong dependence of optimal multicrop water policies on stress sensitivity factors, the maximum yields for each crop and the costs of irrigation and cultivation.

Singh et al. (1995) applied a computer-assisted irrigation scheduling system to optimize water use and enhance crop production for an okra crop in Trinidad and a raspberry field in Quebec. An optimization model called AISSUM (Automatic Irrigation Scheduling System of the University of Montreal) based on a water balance approach was used. AISSUM calculates the water balance and updates the soil moisture storage on a half-hourly basis, and provides a real-time system for irrigation scheduling. The timing of irrigation applications was controlled by integration with rainfall forecasts. The amount of water to be applied, according to the model, is based on the practice of full irrigation. In times of scarcity, deficit strategies could be applied. The sophistication of the technology in AISSUM limits its application to small and often varied farming operations. The model has the potential to be a reliable and cost-effective expert system for managing irrigation scheduling at the farm level. Scheduling between plots has not been provided in this model.

Shyam et al. (1994) developed an optimal operation scheduling model for a canal system using a linear programming technique. The study area was irrigated from the Golawar main canal of Golawar barrage in Uttar Pradesh, India. The poor performance of large irrigation schemes was attributed to inadequate attention being paid to system management and this was one of the major problems in the Golawar command (Bottrall, 1981, Bhuiyan, 1981). Rotational practice was failing to maintain the optimal moisture regime for crops in the command area (Svehlik, 1987). An improved method of water allocation was devised by Shyam et al. (1994). The model was developed to allocate available water in the main canal amongst the secondary and tertiary canals with the objective of maximising the total net return from the command area during the dry season. The objective function was:

$$\text{Maximise } Z = \sum_{i=1}^{I}\sum_{j=1}^{J}(V_{ij}Y_{ij} - C_{ij})X_{ij} - pcD\sum_{n=1}^{N}\sum_{i=1}^{I}\sum_{k=1}^{K}CWD_{nik} - pcB\sum_{n=1}^{N}\sum_{k=1}^{K}CWB_{nk} - pHD\sum_{i=1}^{I}\sum_{k=1}^{K}HD_{nik} - pHB\sum_{n=1}^{N}\sum_{k=1}^{K}HB_{nk} \quad (6.1)$$

where;

V_{ij} = selling price of jth crop in ith tertiary command (Rupees/kg)

Y_{ij} = yield of jth crop in ith tertiary command (kg)

C_{ij} = cost of production of jth crop in ith tertiary command (Rupees/ha)

X_{ij} = area under jth crop in ith tertiary command (ha)

pcB = water transfer cost from main to secondary canals (Rupees/m^3)

pcD = water transfer cost from secondary to tertiary canals (Rupees/m^3)

CWB_{nk} = water to be allocated from the main canal to nth secondary canal during kth period (m^3)

CWD_{nik} = water to be allocated from the nth secondary to the ith tertiary, during kth period (m^3)

pHB = hourly running cost of secondary canals (Rupees/h)

pHD = hourly running cost of tertiary canals (Rupees/h)

HB_{nk}, = the required operation period of the nth secondary canals drawing water from nth branch canal during kth period (h)

HD_{nik} = the required operation period of the ith tertiary canals drawing water from nth branch canal during kth period (h)

The decision variables were area to be allocated under different crop activities in the commands of various tertiaries, water to be allocated from the main canal to the secondaries and the tertiaries, and the required running hours of the secondary canal and the tertiary canals corresponding to available running hours of the main canal. The constraints were, irrigation water supply and operation periods for different canals, canal capacities and minimum allocations to different canals, and the maximum and minimum cropped area restrictions for different tertiary commands. Results were compared for the existing, and three alternative operation plans in which optimal allocation takes place at different stages of water distribution:

- in alternative one, allocation took place at the tertiary level after the existing distribution of water from the main to secondary canals;
- in alternative 2, allocation took place at the tertiary level after area proportionate water allocation from the main to the secondary canals;
- in alternative 3, allocation was done externally and in proportion to command areas of the individual secondary and tertiary canals; an LP model was used to allocate resources optimally at the tertiary level and water allocation to all the minor canals was done in proportion to area.

It was shown that all three operating policies were superior to the existing operating policy, and had a higher total net return. The optimal operating policy in which allocation took place at the secondary canal level had the highest aggregate net return amongst all alternatives.

Reddy et al. (1999) studied irrigation scheduling between canal outlets with different flow capacities. Running times were formulated in a 0-1 linear programming problem. They developed an interactive computer program called ZERO1 to generate the optimal rotation schedule that ensured that all the secondary canals received the specified water allocation during the given rotation. The objective was to schedule the secondary canals to deliver at full supply as far as possible during the given rotation period. The approach to formulating the problem was to derive a schedule which minimizes the differences between required capacity and actual capacity of the supply canal. The objective function was:

$$\text{Minimize } Z = \sum_{k=1}^{N_{level}} C_k Y_k \qquad (6.2)$$

where;

C_k = penalty for exceeding the supply canal capacity by level kth ($)

Y_k = kth level of increase in flow rate capacity of the supply canal

N_{level} = the number of levels of flow rate increases considered

The rotation periods were represented by time blocks of a quarter day up to a maximum of a full day. Parameters defining the problem included: the number of days available during any given rotation period, the number of secondary canals, the flow rate capacity of each secondary canal, and the required running time of each secondary canal in terms of the number of time blocks required. Each canal was permitted to start only once in any time block of the rotation period, and once a secondary was started it was considered to run continuously for its specified time period. A constraint was that in each time block the sum of capacities of secondary canals which received water did not exceed the capacity of the supply canal. Outlets were combined into groups, and these groups of secondary canals were operated together. Appropriate feasible time windows were introduced and secondary canals were not allowed to start outside these time windows. The model was applied to an irrigation project in China - the Xi Le Submain of the Hetao Irrigation project with 117 secondary canals. The results showed that the sum of the secondary canal discharges exceeded the supply canal capacity for some time blocks, a situation that is not feasible in

reality. In practice the secondary canals flows should be flexible within some possible range. However, the use of time blocks and time windows in this study was novel, and a reasonable solution was obtained.

From the review of water scheduling studies, it is clear that a variety of approaches to variants of the scheduling problem have been applied. None of the papers reviewed have addressed the issue of equity in water scheduling, particularly during periods of drought. Approaches based on LP are generally not able to address the issue of equity adequately. Equity in water scheduling is addressed in this thesis.

6.3 Scheduling by the Warabandi System

The warabandi water scheduling system is a system of imposed water scarcity. Warabandi means fixation (*bandi*) of turns (*wara*). The warabandi system was developed nearly a century ago to manage the vast irrigation systems of Northwest India and Southeast Pakistan (over 35 million acres). These systems are designed to operate with limited water supply which must be shared between as many farmers as possible, and protect them against crop failure. Malhotra (1984) has defined the main objectives of the warabandi system as providing:

- high efficiency of water use through the imposition of water scarcity on each user, and
- equity in water use through enforced equal shares of scarce water per unit area among all users.

In warabandi water management, rotation of the available water is set up by turns fixed according to a predetermined schedule. The warabandi water management system has been excellently summarised by Shrestha (1999).

The warabandi system is operated in two tiers. The upper tier comprises primary and secondary canals which are operated by the Irrigation Service (Figure 6.1). The rotational operation of secondary canals depends on the lowest possible supply of the main canal. If the lowest possible supply is one-third of the total capacity of the secondary canals, the secondary canals are formed into three groups with the same or very similar aggregate demand. If the lowest possible supply is one-half of the total capacity of secondary canal, the secondary canals are formed into two groups with the same or similar aggregate demand. Supplies from the main canal are rotated between the groups of secondaries.

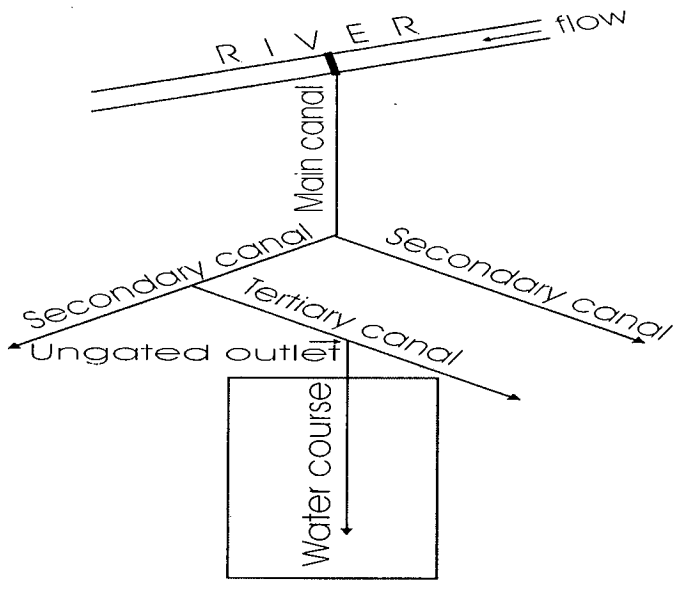

Figure 6.1 *Schematic of warabandi system* (After Malhotra et al. 1984)

The outlets to tertiaries from secondaries are specially designed fixed discharge structures. The design discharge of the outlets is based on their cultivable command area (CCA). When the canal is at full supply level, each tertiary outlet automatically draws its proportional share of water from the secondary canal.

The second tier of rotation is managed by cultivators. During a rotation in the secondaries and tertiaries, which is often seven days, turns are fixed for each land holding or plot. The day, time and duration of supply for each irrigation is specified with supply in proportion to the size of the land holding in the tertiary command area (Malhotra 1984). The turns normally begin at the head and move to the tail of the tertiary. During each turn time, the farmer has the right to use all of discharge flowing in the tertiary (Shrestha 1999).

6.4 Advantages and Limitations of the Warabandi System

The advantages and limitations of the warabandi water management system have been summarised by Shrestha (1999). Although the warabandi system is easy and straightforward for users to understand, theoretically equitable in water distribution, simple in operation and maintenance, and relatively cheap and economic to operate, it has a number of disadvantages (Shrestha 1999):

- it is inflexible in operation;

- it does not attempt to meet crop water requirements in conditions of drought or water scarcity, as supplies are rotated;
- no allowance for seepage losses within tertiaries is included; seepage losses greatly affect the water quantity received by users, and result in uncertainty in the equity of water distribution;
- the warabandi system does not take into account information on soils or crop type;
- turns by holding, can generate severe scheduling problems, especially where holdings are fragmented; also as the water has to flow to different plots during a turn, a substantial water loss can occur;
- in all forms of warabandi, users have to take all water in the tertiary; for large tertiaries, flow rates may be in the range of 2 to 4 m^3/sec and these higher rates are not suitable for good field irrigation, often resulting in inefficient irrigation ;
- secondary canals should run at FSL or should be shut down if the water level in the main canal is below 75% of FSL; ungated outlets are designed as proportional flow distributors so any fluctuation in water level can cause variable flow through these outlets and ultimately the volume of water delivered in a turn will also vary; the system has no compensation for such flow variations.

In the warabandi system, turn times are fixed. It may be possible to improve efficiency and water utilisation with a more sophisticated approach that considers crop requirement and attempts to ensure equity in water supplies to farmers. Development of an objective function for such an approach that could be used in a GA is discussed in the following section.

6.5 An Objective Function for Scheduling

The aim of irrigation scheduling is to apply water before the soil dries below the crop wilting point. Modeling soil moisture should therefore be a fundamental component of scheduling system. In this thesis the approach to soil moisture modeling, and to the determination of irrigation requirements follows that of FAO Irrigation and Drainage Paper 56 (Allen et al. 1998) and is summarised in Appendix D.

A simple irrigation network is shown in Figure 6.2. The objective in scheduling may be considered to be one of optimising water resources utilisation by maintaining soil moisture between field capacity and wilting point, minimising drainage losses. Thus periods of excess

water when soil moisture exceeds field capacity are to be minimised, while maintaining soil moisture above wilting point.

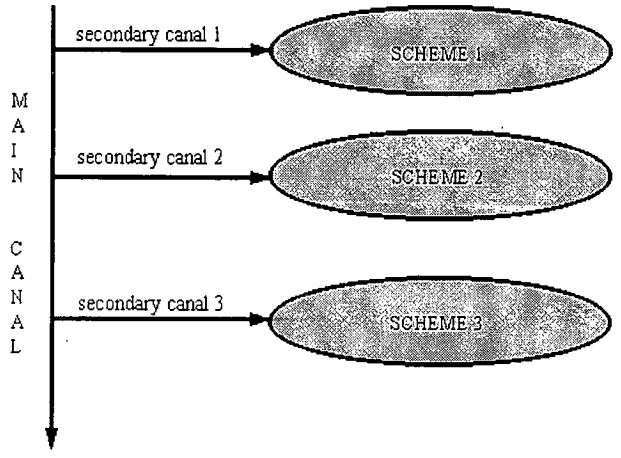

Figure 6.2 A test system

The problem may be described as follows:

$$Minimise\ Z = \sum_{j=1}^{m}\sum_{t=1}^{n} X_{tj} \qquad (6.3)$$

where, X_{tj} = irrigation supply in rotation period t from secondary canal j (mm)

t = time period number

j = scheme or canal number

n = total time periods (days)

m = number of schemes or canals

Subject to: $\theta_{tj} \geq WPtj$ (6.4)

where,

θ_{tj} = soil moisture content in scheme j at time t (mm)

$WPtj$ = moisture content at wilting point in scheme j at time t (mm)

The irrigation supply during any time period may be further defined as follows:

$$X_{tj} = N \cdot x_{tj} \cdot IFLAGtj \qquad (6.5)$$

where,

N = length of time period t (days)

x_{tj} = irrigation full supply rate to scheme j in time period t (mm/day)

$IFLAG_{tj}$= 0 if scheme j is not being irrigated in time period t

= 1 if scheme j is being irrigated in time period t

The irrigation supply may be expressed in terms of the secondary canal capacity:

$$x_{tj} = Q_{jt}/A_j \tag{6.6}$$

where, Q_{jt} = secondary canal supply to scheme j at time period t

A_j = cropped area of scheme j

In addition to the soil moisture constraint given above, canal system capacity constraints have to be considered:

$$\sum_{j=1}^{m} Q_{jt} \leq Qmain_t \tag{6.7}$$

where, $Qmain_t$ = main canal discharge in time period t

For solution by a GA the constraints are incorporated in the objective function as penalties. These may be expressed as follows:

$$\text{If } (\theta_{tj} < WP_{tj}) \qquad P_1 = \sum_{j=1}^{m}\sum_{t=1}^{n}(WP_{tj} - \theta_{tj})^2 \tag{6.8}$$

$$\text{If } \sum_{j=1}^{m} Q_{jt} > Qmain_t \qquad P2 = \sum_{t=1}^{n}(\sum_{j=1}^{m} Q_j - Qmain_t)^2 \tag{6.9}$$

The quadratic form of the penalties increases the sensitivity of the GA to their violation. In addition, penalty factors R_1 and R_2 would be used with them in the evaluation function.

For this formulation, in which secondary canals are assumed to run at their full capacity, the only decision variables are $IFLAG_{tj}$. Schemes either receive irrigation, or don't receive irrigation. In a GA, a gene can thus be represented by a single binary bit.

6.6 Soil Moisture Modelling

Application of the objective function outlined above requires that soil moisture be modeled. For the purposes of this research, a relatively simple soil moisture balance model has been

developed, based on field capacity and wilting point. In any soil layer, moisture content is not permitted to exceed field capacity – any water application in excess of field capacity is assumed to drain to the next soil layer. When soil moisture is between field capacity and wilting point, drainage does not occur, and water is removed from the soil only by evapotranspiration. Wilting point is defined by an allowable depletion factor, which is crop dependent, and is the point at which soil moisture stress starts to occur for a crop. Between wilting point and permanent wilting point (soil moisture tension of 1500 kPa), actual evapotranspiration is assumed to reduce linearly from the potential rate to zero. Details of the approach to soil moisture modelling are given in Appendix D.

Crop evapotranspiration is the principle factor driving soil moisture depletion. The approach to the computation of crop evapotranspiration is based on the methods outlined by Allen et al. (1998) in FAO Irrigation and Drainage Paper No 56. A dual crop coefficient approach has been adopted to account for water stress periods and resulting reductions in evapotranspiration.

The constraints given in equation 6.8 is applied to the crop root zone only. Crop root development must therefore be modeled through the growing period. The recommendations of Allen et al. (1998) have again been followed for this. Root zone soil moisture is tracked and adjusted as the roots develop. For modelling purposes, the soil column is divided into a series of discrete layers, in each of which soil moisture is tracked. As the root zone develops, encompassing new layers, soil moisture in the root zone is re-adjusted.

It would be preferable to model soil moisture movements using a model such as WAVE (Vanclooster et al. 1996) in which the Richards equations are solved. The incorporation of this type of approach was beyond the scope of the present research, however. The simple model adopted is adequate to permit evaluation of a GA approach to water scheduling, which is the main objective.

6.7 Preliminary Testing of the GA Scheduling Model

6.7.1 System and Crop Characteristics

The GA has been applied to the very simple test system shown in Figure 6.2. The test system comprises 3 schemes in which irrigation water is distributed by secondary canals. Each scheme has an area of 100 ha. Flow in the main canal is continuous, but secondary

canals are rotated and operated at full capacity. The physical characteristics assumed for the system are shown in Table 6.1. The main canal capacity was set to 0.24 m^3/sec, and secondary canal capacity to 0.12 m^3/sec each. Fields were assumed to be 500 m long and the secondary canal length was thus taken to be 2000 m. Losses in opening and closing canals were taken to be 1500 m^3, or 1.5 mm. The inclusion of these losses is important as it is through these that the GA is forced to limit the frequency of canal operation. With the basic set up given in Table 6.1, there is no water stress induced in the system. A water stress situation was introduced by reducing the main canal capacity to 0.14 m^3/s, and that of the secondary canals to 0.07 m^3/s.

Table 6.1 Specified criteria of the test system

Items	Specified Data		
scheme number	1	2	3
scheme area (ha)	100	100	100
FC (m^3m^{-3}) for sandy soil	0.17	0.17	0.17
PWP (m^3m^{-3}) for sandy soil	0.07	0.07	0.07
secondary canal full supply (m^3/sec)	0.12	0.12	0.12
main canal full supply (m^3/sec)	0.24		
irrigation efficiency	70%		

A crop of beans was assumed for all schemes with a growing period of 100 days. The length of development stages for beans are summarised in Table 6.2. Planting in schemes 1 and 2 was assumed to occur on the same day (1st June), but planting on scheme 3 was staggered by 3 days. Potential crop evapotranspiration, ET_c has been calculated throughout the growing period, on the basis of ET_o values for Chiengrai in Thailand. Crop characteristics were taken from Allen et al. (1998). The minimum and maximum rooting depths were assumed to be 0.15 m and 0.80 m respectively, and the soil moisture depletion fraction taken to be 0.45. Potential crop evapotranspiration is summarised in Table 6.3. Initial SMC was set at wilting point in all schemes.

Table 6.2 Length of development stages for beans

Days 1-25	Days 26-50	Days 51-80	Days 81-100	Seasonal Total (days)
25	25	30	20	100

Table 6.3 Potential crop evapotranspiration (mm/month)

Jan	Feb	Mar	Apr	May	June	July	Aug	Sep	Oct	Nov	Dec
93	125	151	169	158	143	135	124	128	122	105	91

In computing crop water requirements, it has been assumed that there is no effective rainfall. Crop water requirements have been computed for each day in the growing period.

6.7.2 Representation Schemes

In this test system, genes in a chromosome represent the decision variables ($IFLAG_{ti}$), t = $1,m$, i = 1, n, where m is the number of time steps and n is the number of schemes. In this preliminary test system, a time step of 1 day was taken. With three schemes, the total chromosome length was thus 300. A chromosome may represent the decision variable in two ways. One is to group the genes by scheme, as shown in Figure 6.3, and the other is to group genes by time step. The former approach has been used in preliminary testing. Further discussion of other formulations is presented in Chapter 7.

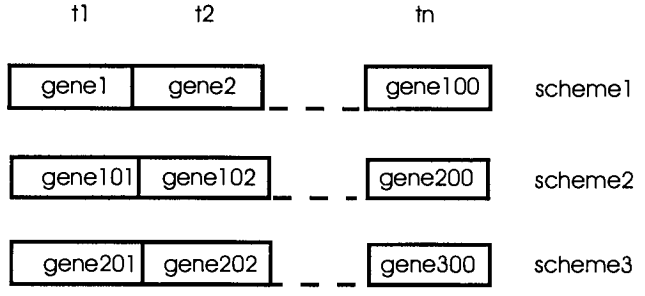

Figure 6.3 Decision variable grouping by scheme

The representation scheme, although binary, is in effect real valued in that a single value represents each gene. It is simplified in that each of these values are binary, taking on the value of 0 or 1.

6.7.3 Operators Used

The GA was set up with uniform crossover, tournament selection, and conventional mutation, as decision variables were only switched between 0 and 1. A population size of

100 was adopted with a probability of crossover of 0.85, and 0.05 mutations per chromosome. The crossover and mutation probabilities were determined through a series of sensitivity runs. The number of mutations per chromosome is very much lower than found in the water allocation problem or in the reservoir problems studied by Wardlaw and Sharif (1999).

6.7.4 Scheduling Results With No Water Stress

With no water stress in the system sensitivity analysis indicated that the appropriate penalty factor for the wilting point constraint (R1) was 3, and that for capacity constraint (R2) was 12500. The very high value of R2 is required because of differences in the orders of magnitude of the constraint parameters.

With no water stress, canal capacity and wilting point constraints were satisfied in all time periods. The best fitness was 303 mm, and this of course is the water supply to the system in a 100-day season - an average of 303 mm per scheme. The crop water requirements for three schemes were 302 mm, 302 and 300 mm for scheme 1, 2, 3 respectively. The lower value for scheme 3 reflects the later planting date. The soil profile was assumed to be at wilting point at the start of the growing period, and as a result, there is a very close match between water supplied and water required, and there is in effect no over application. Table 6.4 shows the scheme water balances and Figure 6.4 shows the irrigation schedule produced by the GA. Shading area represents the irrigation period, while blank block represents the non-irrigation period. As can be seen from the schedule, irrigation periods lengthen as the crop develops. Variations in soil moisture content throughout the irrigation period are shown in Figure 6.5. The GA has successfully maintained SMC between wilting point and field capacity throughout the growing period.

Table 6.4 Scheme water balances (non-stress case)

Total amount (mm)	Scheme 1	Scheme 2	Scheme 3
Irrigation	304	304	301
ETc	302	302	300
Ea	302	302	300
Change in storage	-1.177	-1.420	-0.580
Drainage	0	0	0

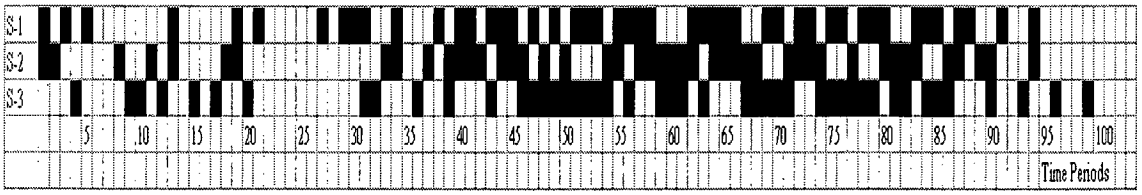

Figure 6.4 Irrigation schedule generated by GA, no water stress

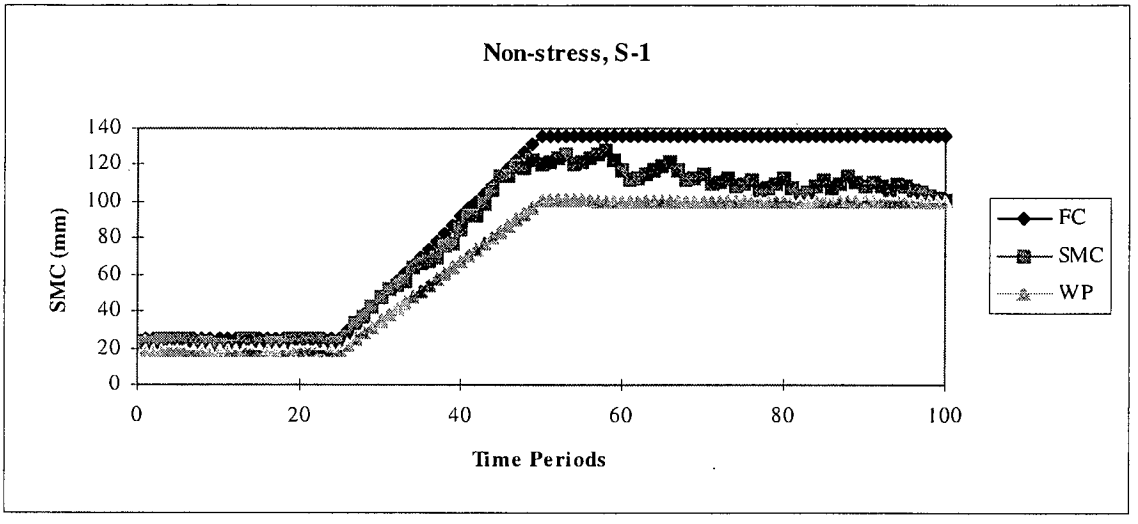

Figure. 6.5 Simulated root zone soil moisture content, no water stress

6.7.5 Results With Water Stress

As indicated earlier, water stress was introduced to the system by reducing the main canal capacity to 0.14 m^3/s, and the secondary canal capacities to 0.07 m^3/s. Cropping and soil characteristics were assumed to be as given in Tables 6.1 and 6.2.

With water stress in the system, sensitivity analysis indicated that the best fitness would be achieved with a crossover probability was 0.85, and with 0.05 mutations per chromosome. The number of mutations per chromosome are as same as in the case of no water stress, but still much lower than for other problems. Penalty factor for wilting point constraint (R1) was 1 and penalty factor for capacity constraint (R2) was 10000.

Under water stress conditions, the water supply to the system in 100-day season was an average of 241 mm per scheme. Scheme water balances are summarised in Table 6.5. There are minor differences in the actual evapotranspiration from the different schemes, but

the ratio of Ea/ETc is similar in all schemes, indicating that equity in water distribution has been preserved.

The irrigation schedule produced by the GA is shown in Figure 6.6. Figure 6.7 shows the simulated root zone soil moisture content. With water stress it is not possible to satisfy the wilting point constraint of equation 6.8, and this is used to maintain equity in water allocation between the schemes. The total cumulative water stress of 732 mm days in scheme 1, 616 mm days in scheme 2 and 626 mm days in scheme 3. The higher stress in scheme 1 is reflected in the lower actual evapotranspiration from that scheme. It is possible that improved equity in cumulative stress could have been achieved through further modification of the penalty factors.

Table 6.5 Scheme water balances (water stress case, all values in mm)

Total amount (mm)	Scheme 1	Scheme 2	Scheme 3
Irrigation	231	251	240
ETc	302	302	300
Ea	241	245	238
Ea/ETc	0.80	0.81	0.79
Change in storage	11.428	1.239	0.258
Drainage	1.458	7.780	2.352

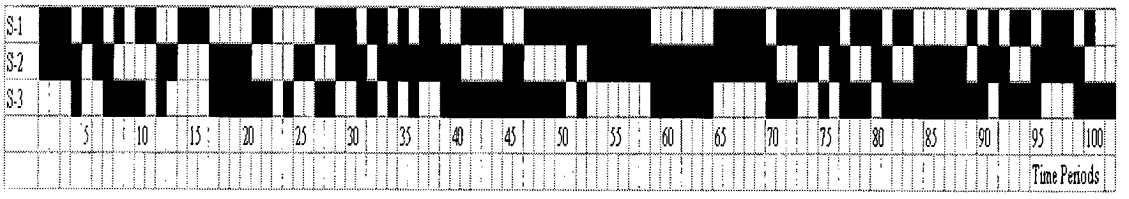

Figure 6.6 Irrigation schedule generated by GA with water stress

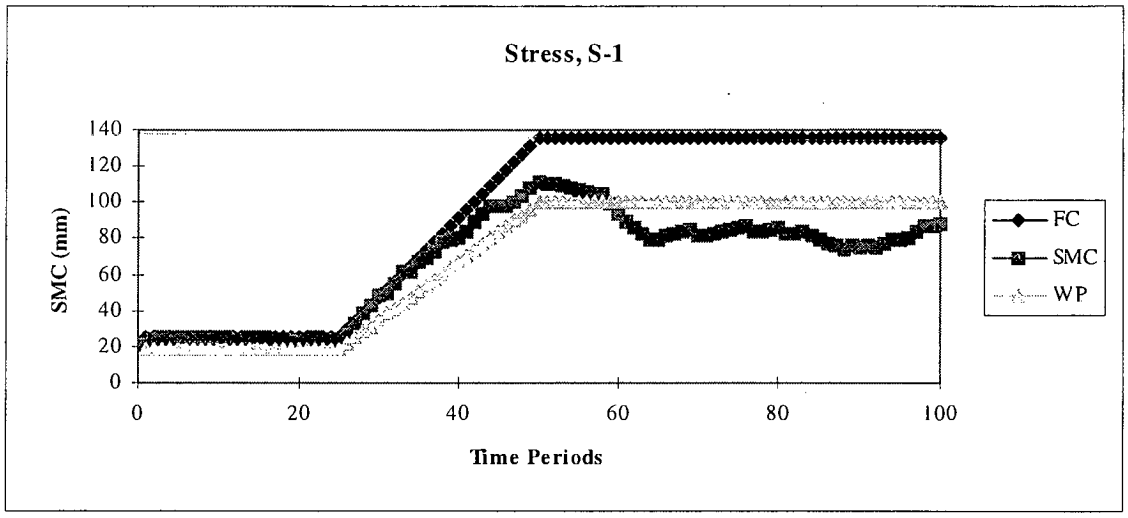

Figure 6.7 Simulated root zone soil moisture content with water stress

6.7.6 Comparison of Stress and Non-Stress Condition

A comparison of scheduling results between soil water stress condition and non-stress conditions has been made. The total water delivery with no water stress was 303 mm, while with water stress conditions it was 241 mm. In the water stress case, irrigation frequency was reduced, thereby reducing the losses associated with canal opening and closing. The total number of irrigations per scheme was reduced from 22 in the non-stress case to 18 in the stress case. The number of irrigation days per scheme was increased from 46 in the non-stress case to 63 in the stress case. Table 6.6 summarises the comparison of both cases.

Table 6.6 Comparison of soil water stress and non-stress case

Items	Non-stress			Stress		
Avg. supply water (mm/scheme)	303			241		
Cumulative over supply (mm)	0.000			11.6		
Average cumulative stress (mm days)	0.000			658		
Secondary canal	S-1	S-2	S-3	S-1	S-2	S-3
Irrigation days (days)	47	46	46	62	65	63
Irrigation frequency (times)	25	20	22	21	16	18

6.7.7 Results With 12 hr Time Period

The influence of the length of time period used in the GA was investigated by reducing the time period from 1 day to 12 hours on the water stress configuration. Under the 12 hour time step, the total water supply was 223 mm, compared with 241 mm on a 1 day time period. The cumulative soil moisture deficits below wilting point amounted to an average of 801 mm days per scheme compared with 658 mm days per scheme on a 1 day time period. The shorter time period did not result in any improvement in either equity or total water supplied. The irrigation schedule produced by the GA is shown in Figure 6.8. Scheme soil moisture balances are summarised in Table 6.7, and variations in soil moisture content are shown in Figure 6.9.

Under the 12 hour time step, the GA has operated the secondary canals more frequently, resulting in higher losses than with the 1 day time step. In addition, the gene length is twice that of the 1 day time step, and this makes it more difficult to preserve good solutions, as well as resulting in longer execution times.

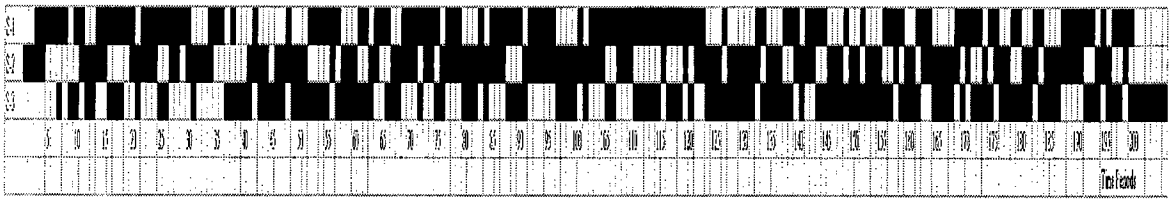

Figure 6.8 Simulated root zone soil moisture content with water stress, 12 hour time periods

Table 6.7 Water balance by scheme (12-hr time period)

Total amount (mm)	Scheme 1	Scheme 2	Scheme 3
Irrigation	217	226	225
ETc	302	302	300
Ea	224	236	224
Change in storage	11.091	10.111	0.865
Drainage	4.472	0.206	1.962

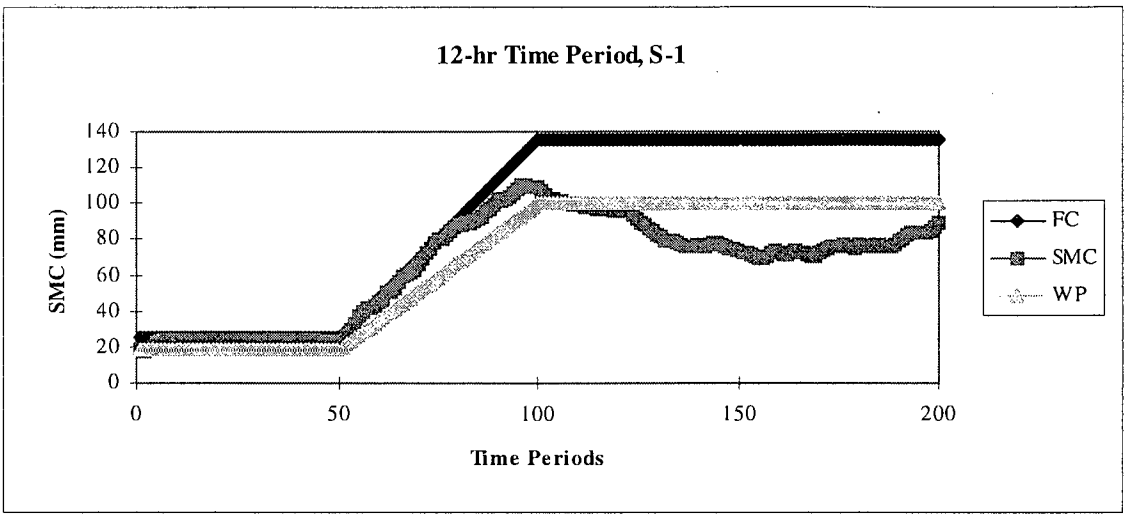

Figure 6.9 Simulated root zone soil moisture content, 12 hour time periods

A comparison of key results with a 12 hour and 1 day time period is shown in Table 6.8. There is no advantage in using 12 hour time periods.

Table 6.8 Comparison of results with 1-day and 12 hr time periods

Items	1 day T-Period			12 hr T-Period		
Avg. supply water (mm/scheme)	241			223		
Cumulative over supply (mm)	12			7		
Cumulative stress (mm.days)	658			801		
Secondary canal	S-1	S-2	S-3	S-1	S-2	S-3
Irrigation days (days)	62	65	63	63	66	66
Irrigations frequency (times)	20	16	18	32	34	35
Avg. exe. time (minutes/100 generations)	0.27			0.89		

6.7.8 Conclusions from Preliminary Testing

From the test system, it is clear that the developed GA using what is effectively a Zero-1 decision criteria, is capable of solving the water scheduling problem, as defined, efficiently and effectively. These results justified the exploration of GA applications to more complex and practical scheduling situations, and to alternative scheduling approaches. These are discussed in the Chapter 7.

6.8 Sensitivity Analyses

Sensitivity analysis was conducted on the water stress case using a 1 day time period. It was found that 0.05 mutations per chromosome produced the best fitness as shown in Figure 6.10. The crossover that produced the best fitness was at 0.85 as shown in Figure 6.11. The GA is quite robust in that good results can be obtained over a fairly wide range of mutation and crossover probabilities.

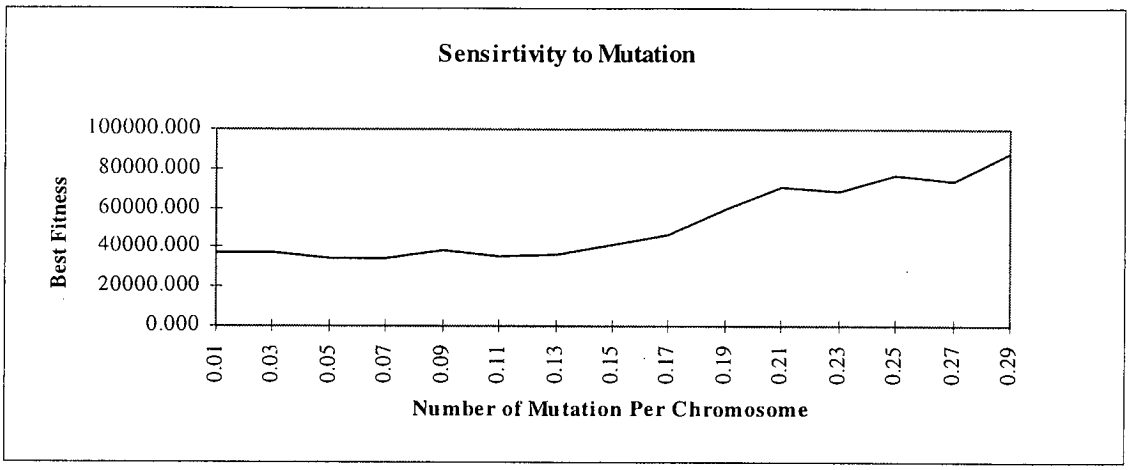

Figure 6.10 Sensitivity to mutation

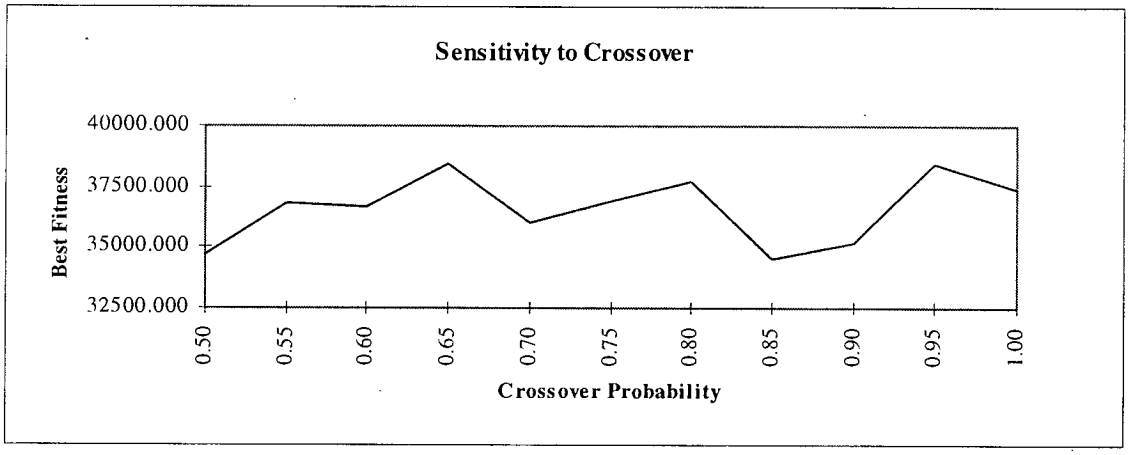

Figure 6.11 Sensitivity to crossover

7. APPLICATION OF A GA IN ALTERNATIVE SCHEDULING APPROACHES

7.1 General

In the preliminary GA application to water scheduling outlined in Chapter 6, only one approach to scheduling was considered. That approach may be termed the Zero-1 approach. It is possible to set up a GA to tackle other approaches. Both the Warabandi approach, and what is referred to as a running time approach, have been investigated through application to a more complex, although still hypothetical, network. The fundamental objective of each approach remains the same, but each approach is set up with different decision variables. In this chapter evaluation of these alternative approaches is presented. A practical application of a GA to water scheduling is also presented.

7.2 The More Complex Network for Scheduling

A more complex irrigation network has been devised in which rotation is required between secondary canals, and between tertiary canals within secondary units. The system comprises a main canal with four secondary canals. Each secondary canal comprises four tertiary canals as shown in Figure 7.1. Investigations have been carried out with a main canal capacity of 0.28 m^3/s. Secondary canal capacities were fixed at 0.14 m^3/s, and tertiary canal capacities at 0.07 m^3/s. Table 7.1 summarises the assumed system characteristics. Fields were assumed to be 500 m long and the canal lengths thus taken to be 2000 m. Losses in opening and closing tertiary canals were taken as before to be 1500 m^3. Losses in opening and closing secondary canals were taken to be 6000 m^3.

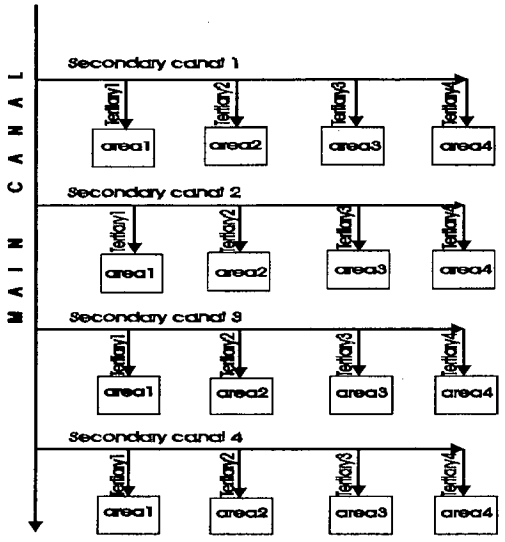

Figure 7.1 The more complex test network for scheduling

Table 7.1 Specified criteria of a more complex system

Items	Specified Data															
Total command area (ha)	1600															
Secondary number	S1				S2				S3				S4			
Secondary command area (ha)	400				400				400				400			
Tertiary number	T1				T2				T3				T4			
Tertiary command area (ha)	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
FC (m^3m^{-3})	0.17				0.17				0.17				0.17			
PWP (m^3m^{-3})	0.07				0.07				0.07				0.07			
Main canal full supply (m^3/sec)	0.28															
Secondary full supply (m^3/sec)	0.14				0.14				0.14				0.14			
Tertiary full supply (m^3/sec)	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07
Irrigation efficiency	70%															

As in the test network used in Chapter 6, a bean crop was assumed for which minimum and maximum rooting depths were 0.15 m and 0.80 m respectively and the allowable soil moisture depletion fraction was 0.45. Effective rainfall was set to zero in all time periods. It was assumed that pairs of tertiaries would be planted together, but that there would be 5 days of stagger between adjacent pairs. Tertiaries 1 and 2 of secondaries 1 and 2 were assumed to be planted together. Tertiaries 3 and 4 of secondaries 1 and 2 were planted together, but 5 days after tertiaries 1 and 2.

In the same manners, tertiaries 1 and 2 of secondaries 3 and 4 were assumed to be planted together, but 5 days after tertiaries 3 and 4 of secondaries 1 and 2. Tertiaries 3 and 4 of secondaries 3 and 4 were planted 5 days after tertiaries 1 and 2. Crop water requirements were calculated as before.

7.3 Revised Objective Function and Constraints

7.3.1 General

For this more complex system it was necessary to re-dimension the objective function so that supplies could be referenced by secondary and tertiary canal. The objective or evaluation function was expressed as follows:

$$Minimise\ Z = \sum_{t=1}^{n}\sum_{j=1}^{m}\sum_{i=1}^{l} X_{ijt} + R1.P1 + R2.P2 + R3.P3 \qquad (7.1)$$

where: X_{ijt} = irrigation supply to tertiary i of secondary j in time period t,

i = tertiary canal number

j = secondary canal number

t = time step

l = number of tertiary canals

m = number of secondary canals

n = number of time periods

$P1$, $P2$ and $P3$ are penalties for constraint violation, and $R1$, $R2$ and $R3$ are the penalty weighting factors.

The crop stress penalty P1 was common to all scheduling approaches and was expressed as follows:

$$PI_{ijt} = 0$$

$$If\ \theta_{ijt} < WP_{ij}\ ,\ PI_{ijt} = (WP_{ij} - \theta_{ijt})^2$$

$$PI = \sum_{t=1}^{n}\sum_{j=1}^{m}\sum_{i=1}^{l} PI_{ijt} \qquad (7.2)$$

where,

WP_{ij} = wilting point in tertiary i of secondary j

θ_{ijt} = root zone soil moisture content in tertiary i of secondary j in time period t

The decision variables and the expressions for canal capacity constraints varied depending upon the scheduling approach being adopted, and are outlined below.

7.3.2 Decision Variable and Capacity Constrains for Zero-1 Approach

The Zero-1 approach was outlined in Chapter 6, and only dimensions change in application to this more complex problem. For this more complex problem the total number of decision variables is 1600. The irrigation supply to any scheme is defined as follows:

$$X_{ijt} = q_{ij}.IFLAG_{ijt} \qquad (7.3)$$

where, q_{ij} = full supply capacity of tertiary canal i of secondary canal j

$IFLAG_{ijt}$ = takes a value of zero or one, indicating whether or not there is irrigation in time step t, and is the decision variable

The canal capacity constraints are expressed as follows:

i) The secondary canal capacity penalty:

$$P2_{jt} = 0$$

$$\text{If in any time } t \sum_{i=1}^{l}(q_{ij}.IFLAG_{ijt}) > Q_j\ ,\ P2_{jt} = \left(\sum_{i=1}^{l} q_{ij}.IFLAG_{ijt} - Q_j\right)^2$$

$$P2 = \sum_{t=1}^{n}\sum_{j=1}^{m} P2_{jt} \qquad (7.4)$$

where: Q_j = full supply capacity of secondary canal j

ii) The main canal capacity penalty:

$$IFLG\_S_{jt} = 0$$

$$\text{If, } \sum_{i=1}^{I} IFLAG_{ijt} \rangle 0 \ , IFLG\_S_{jt} = 1$$

$$P3_t = 0$$

$$\text{If } \sum_{j=1}^{m} (Q_j . IFLG\_S_{jt}) > Qmain_t, \ P3_t = \left(\sum_{j=1}^{m} Q_j . IFLG\_S_{jt} - Qmain_t \right)^2$$

$$P3 = \sum_{t=1}^{n} P3_t \qquad (7.5)$$

where, $Qmain_t$ = flow in main canal in time step t

The above canal capacity constraints will in effect control the opening and closing combinations of canals.

7.3.3 Decision Variable and Capacity Constraints for Warabandi Approach

In the warabandi approach, the duration for which schemes receive water is fixed, and the interval between irrigations is fixed, although either parameter may vary between schemes as command areas differ. There are therefore two primary decision variables per scheme. It is necessary also to define the starting time period for irrigation in each scheme. The starting time may also be incorporated in the GA as a decision variable, and this is desirable as in complex systems the best combination of starting times will be difficult to define, and arguably this would be the objective of applying optimisation. For the test network considered here, the total number of decision variables is thus 48:- 16 starting times, 16 irrigation durations, and 16 intervals between irrigations. Generally one would expect these variables to be expressed in units of days, and it is possible in a GA to represent them as real values. A chromosome thus comprises 48 genes. The decision variables have been defined as follows:

SI_{ij} = starting time step for irrigation in tertiary i of secondary j

DI_{ij} = duration of irrigation in tertiary i of secondary j

II_{ij} = interval between irrigations in tertiary i of secondary j

For each candidate solution or chromosome in the GA population defined by the above decision variables, the variable $IFLAG_{ijt}$ can easily be determined. It is thus possible to use the same canal capacity constraints as outlined in equations 7.4 to 7.5 without modification. The irrigation supply is computed from equation 7.3.

7.3.3.1 Decision Variables and Capacity Constraints for Running Time Approach

The running time approach is a variant of the warabandi approach, in which the time periods in which the tertiary canals were opened, TPO_{ijk}, and the running time during each opening, RTO_{ijk}, were defined as decision variables. This approach was perceived to have the potential to overcome the restrictions imposed by the warabandi approach, and to more closely match supplies with demands. If these two variables are defined throughout the growing period then the variable $IFLAG_{ijk}$ can again be very easily determined, as in the warabandi approach. The same evaluation function and constraints outlined in equations 7.1 to 7.5 can then be applied unaltered. There are difficulties in formulating this approach for solution by GA, however. The problem is that for any scheme, the variables TPO_{ijk} must increase progressively with time. This is difficult to achieve through the random generation process of a GA, and has required the inclusion of additional constraints to ensure that progression was maintained in the value of starting time steps. When populations were generated initially, the randomly generated values of TPO were put into rank order, and in subsequent GA operations, chromosomes which did not maintain progression were not permitted into the next generation.

It was found that solutions could not be obtained that satisfied constraint and covergence criteria. The main reason for this is thought to have been the requirement for progression in TPO. It had been thought that the approach would have been more effective than the Zero-1 approach, requiring fewer genes. This proved not to be the case, and formulation of the running time approach by GA is not possible. An evaluation of the Zero-1 and Warabandi scheduling approaches with GA is presented in the following section.

7.4 Evaluation of Scheduling Results

7.4.1 Results with Zero-1 Criteria

The zero-1 approach was applied with a population size of 100, a crossover probability of 0.9, and 0.09 mutations per chromosome. The convergence criteria set were that the improvement in minimum fitness over a minimum generation gap of 300 must be less than 0.001%, and that the canal capacity constraints must be satisfied to within a tolerance of ± 0.1% of the canal capacity. On the basis of the above criteria, convergence was obtained in 2500 generations.

The irrigation schedules obtained are presented in Figures 7.2 and 7.3, for tertiary and secondary canals respectively. Variations in soil moisture for a typical tertiary are shown in Figure 7.4, and cumulative soil moisture stress, expressed in mm days, are presented in Figure 7.5 for all tertiaries. Irrigation frequency to each tertiary canal is presented in Figure 7.6. Total irrigation supplies to each of the tertiaries are presented in Table 7.2.

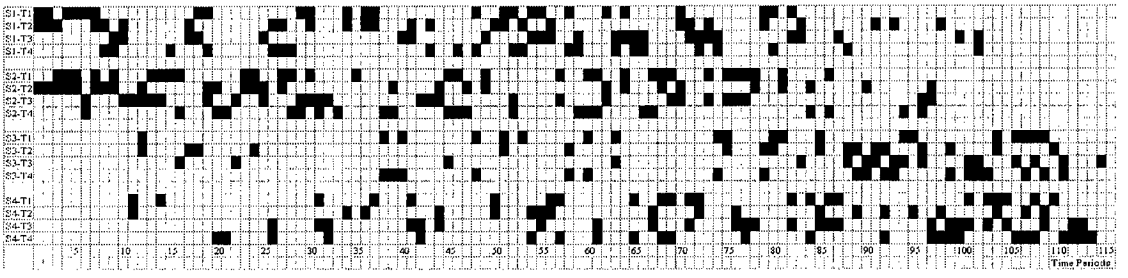

Figure 7.2 Irrigation schedule in tertiaries with Zero-1 criteria

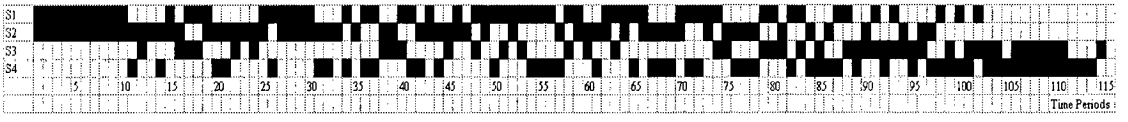

Figure 7.3 Irrigation schedule in secondaries with Zero-1 criteria

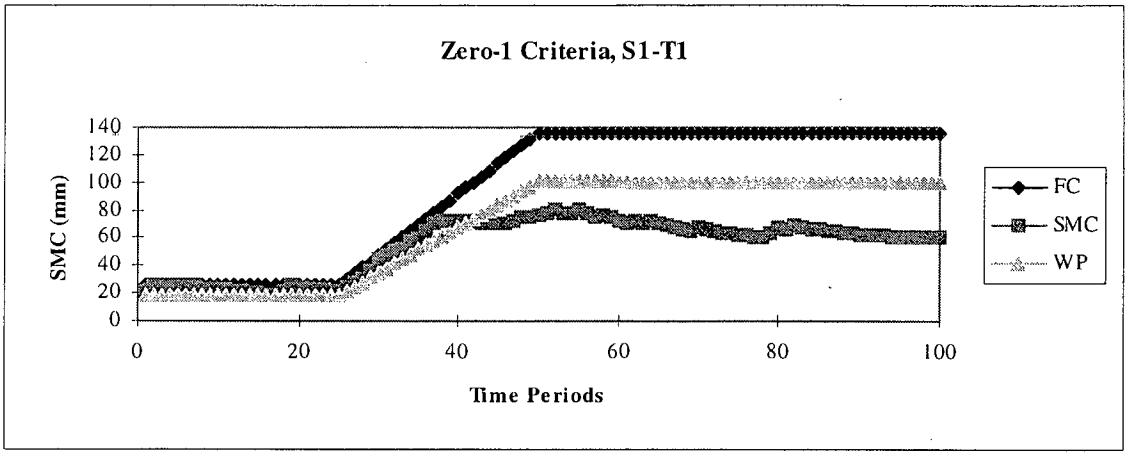

Figure 7.4 Simulated root zone soil moisture content with Zero-1 criteria

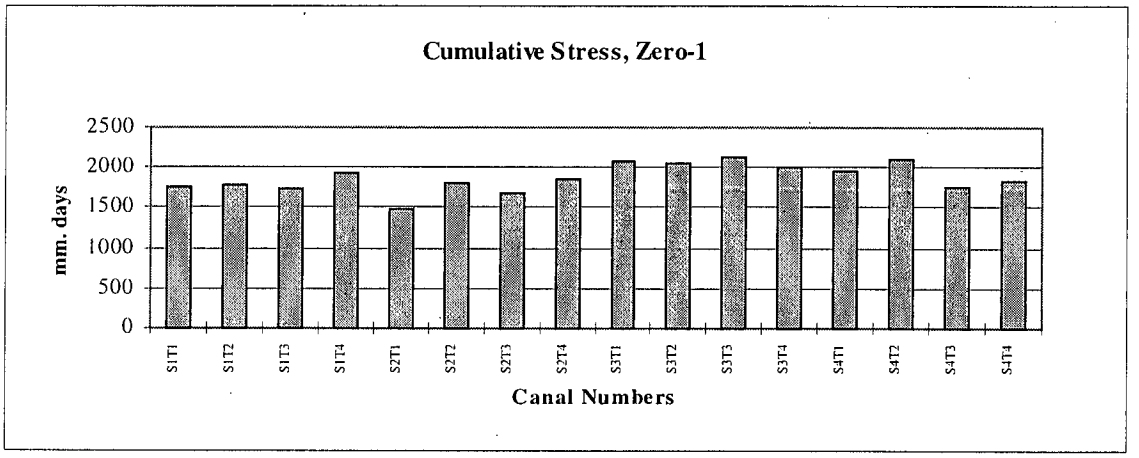

Figure 7.5 Cumulative stress with Zero-1 approach

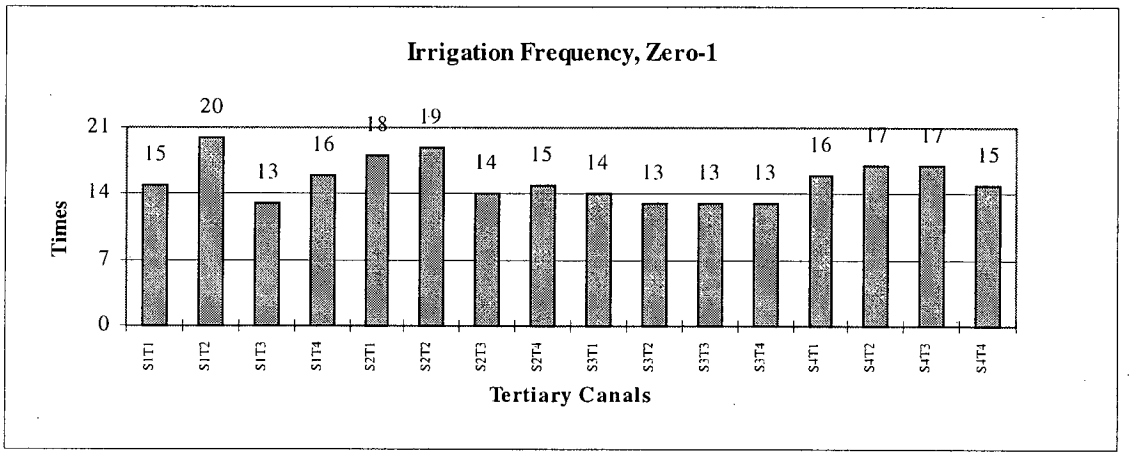

Figure 7.6 Irrigation frequency with Zero-1 approach

Table 7.2 Water balance with zero-1 criteria (mm)

Secondary 1	**S1-T1**	**S1-T2**	**S1-T3**	**S1-T4**
Irrigation	104	107	107	87
ETc	302	302	299	299
Ea	144	138	140	125
Ea/ETc	0.475	0.458	0.468	0.419
Change in storage	-40	-32	-33	-38
Drainage	0.000	0.000	0.000	0.000
Secondary 2	**S2-T1**	**S2-T2**	**S2-T3**	**S2-T4**
Irrigation	135	113	116	94
ETc	302	302	299	299
Ea	171	137	151	131
Ea/ETc	0.565	0.453	0.504	0.437
Change in storage	-36	-34	-36	-37
Drainage	0.000	10.463	0.508	0.000
Secondary 3	**S3-T1**	**S3-T2**	**S3-T3**	**S3-T4**
Irrigation	80	70	65	77
ETc	295	295	292	292
Ea	98	97	87	99
Ea/ETc	0.332	0.330	0.298	0.341
Change in storage	-18	-27	-22	-23
Drainage	0.000	0.000	0.000	0.000
Secondary 4	**S4-T1**	**S4-T2**	**S4-T3**	**S4-T4**
Irrigation	91	71	106	104
ETc	295	295	292	292
Ea	111	98	125	111
Ea/ETc	0.376	0.332	0.428	0.381
Change in storage	-20	-27	-19	-7
Drainage	0.000	0.000	0.000	0.000

The GA has been effective in reaching a solution to the problem, although the equity achieved was not as good as had been hoped. From Table 7.2 it will be noted that there are

differences in the Ea/ETc ratios between secondary units. The reason for this is the complexity of the penalty function. Both canal capacity and soil moisture constraints should be satisfied. The test considered is one of very severe water stress, and performance on less severe stress cases would be expected to be better. In this case the average Ea/ETc ratio was 0.412, and the standard deviation 0.07.

Further model runs were carried out with reduced water stress condition. Figure 7.7 shows variability in the Ea/ETc ratios as the water stress is reduced through enlarging the canal capacities. With reduced stress it is possible to achieve better equity in the system. Under what is effectively a zero stress situation, the average Ea/ETc ratio increases to 0.974, and the standard deviation reduces to 0.02.

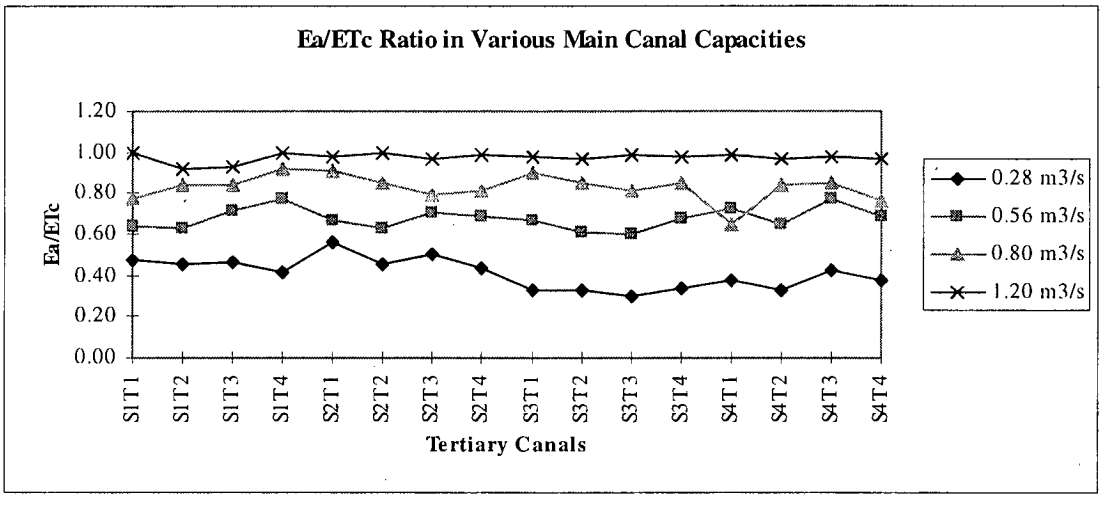

Figure 7.7 Ea/ETc ratios under different water stress conditions

7.4.2 Results With Warabandi Approach

The Warabandi approach was applied with a population size of 100, a crossover probability of 0.9, and 0.3 mutations per chromosome. The convergence criteria were the same as with Zero-1 criteria. Convergence was not improved after 900 generations. The irrigation schedules obtained are presented in Figures 7.8 to 7.9, for tertiary and secondary canals. Variations in soil moisture for a typical tertiary are shown in Figure 7.10, and cumulative soil moisture stress, expressed in mm days, are presented in Figure 7.11 for all tertiaries. Irrigation frequency to each tertiary canals is presented in Figure 7.12. Total irrigation supplies to each of the tertiaries are presented in Table 7.3.

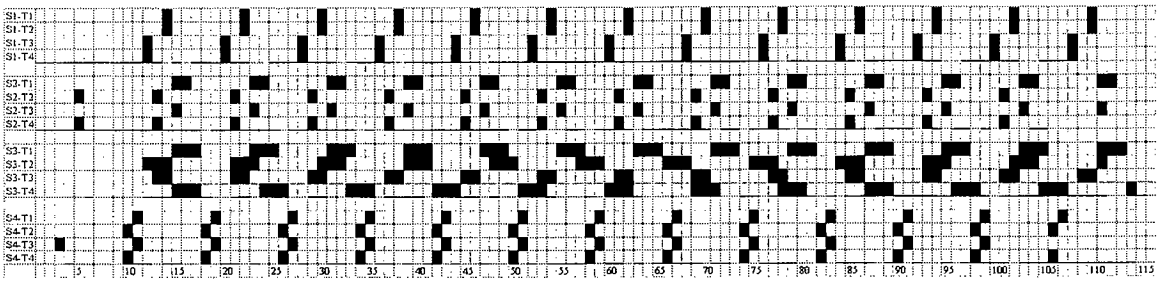

Figure 7.8 Irrigation schedule in tertiaries with Warabandi approach

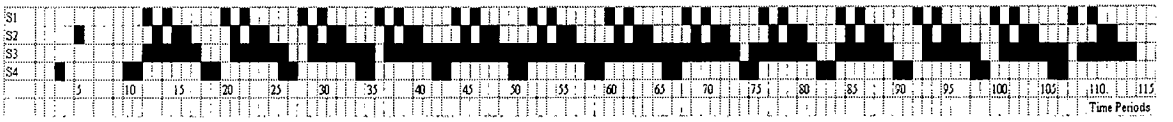

Figure 7.9 Irrigation schedule in secondaries with Warabandi approach

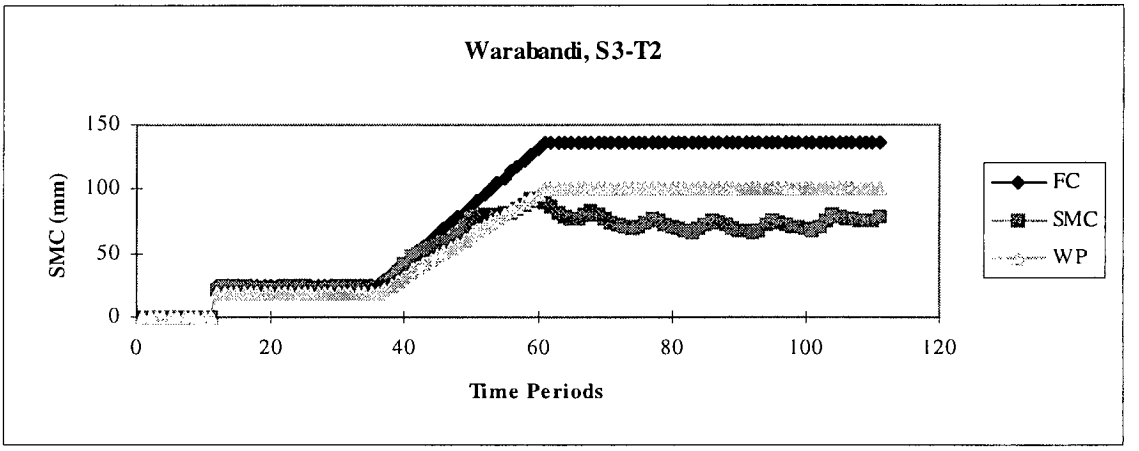

Figure 7.10 Simulated root zone soil moisture content, Warabandi approach

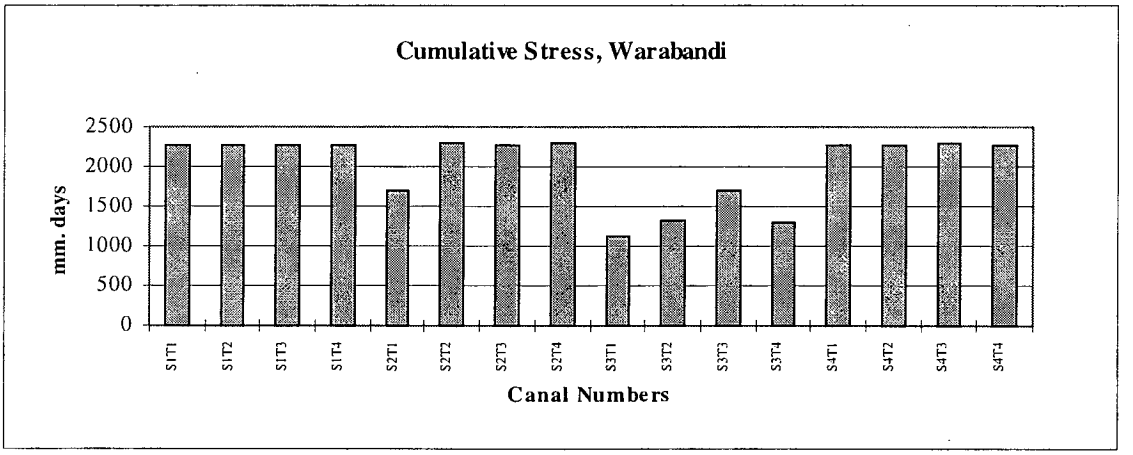

Figure 7.11 Cumulative stress, Warabandi approach

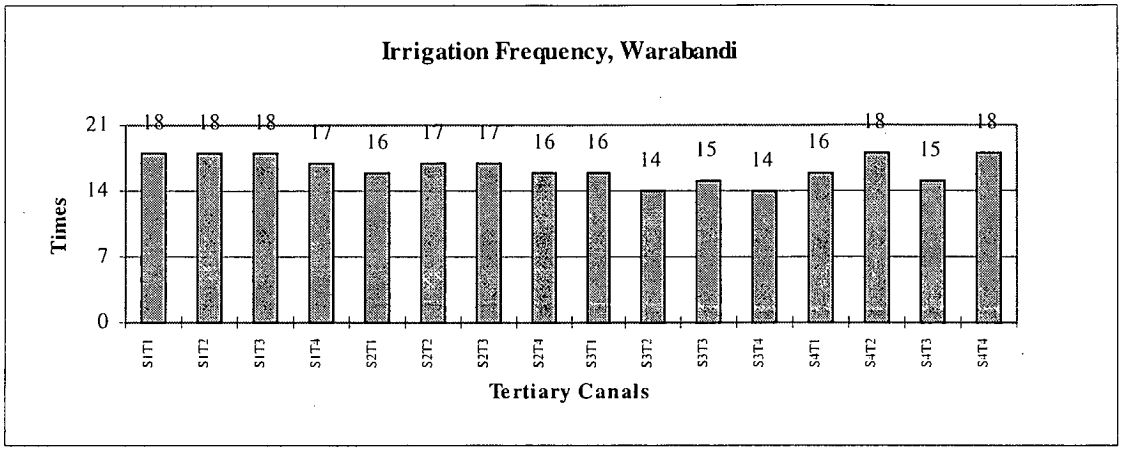

Figure 7.12 Irrigation frequency, Warabandi approach

The GA has been successful in creating an irrigation schedule under the Warabandi approach that satisfies all of the system constraints. The schedule is not as equitable as that produced by the Zero-1 approach. Fixed times of delivery, irrespective of water need, results in greater water stress. With the Warabandi approach, the average Ea/ETc ratio was 0.367, and the standard deviation in Ea/ETc was 0.14. The performance was thus a lot poorer than that of the Zero-1 approach.

The schedule produced under the Warabandi approach is presented in Table 7.4. As evident from Figure 7.9, secondary 3 receives significantly more water than the other canals. It was thought that this may to some extent have been influenced by the requirement to minimise supplies as part of the objective function, and that during water stress periods, this may be its conflict with the requirement to minimise water stress. This issue was investigated by removing the supply from the objective function. This had no impact. The weight applied to the water stress penalty over-rides the water supply criteria.

With less water stress in the system, the Warabandi approach does perform slightly better in terms of equity, in much the same way as did the Zero-1 approach. However, even under what is effectively a zero stress situation, the original formulation of the Warabandi approach only produced an Ea/ETc ratio of 0.815, compared to 0.974 with the Zero-1 approach. The reason for this is the influence of the supply minimisation objective in the early part of the schedule resulting in longer irrigation intervals and lower irrigation applications than are required in the latter part of the schedule.

Table 7. 3 Water balance with warabandi approach (mm)

Secondary 1	**S1-T1**	**S1-T2**	**S1-T3**	**S1-T4**
Irrigation	46	46	46	46
ETc	293	293	295	295
Ea	83	83	83	83
Ea/ETc	0.283	0.283	0.281	0.281
Change in storage	-37	-37	-37	-37
Drainage	0.000	0.000	0.000	0.000
Secondary 2	**S2-T1**	**S2-T2**	**S2-T3**	**S2-T4**
Irrigation	112	46	46	46
ETc	293	300	293	300
Ea	138	83	83	83
Ea/ETc	0.470	0.277	0.283	0.277
Change in storage	-26	-37	-37	-37
Drainage	0.000	0.000	0.000	0.000
Secondary 3	**S3-T1**	**S3-T2**	**S3-T3**	**S3-T4**
Irrigation	178	154	112	154
ETc	293	295	294	293
Ea	192	176	138	176
Ea/ETc	0.655	0.597	0.468	0.601
Change in storage	-14	-22	-26	-22
Drainage	0.000	0.000	0.000	0.000
Secondary 4	**S4-T1**	**S4-T2**	**S4-T3**	**S4-T4**
Irrigation	46	46	46	46
ETc	295	296	301	296
Ea	83	83	83	83
Ea/ETc	0.281	0.280	0.276	0.280
Change in storage	-37	-37	-37	-37
Drainage	0.000	0.000	0.000	0.000

Table 7.4 Scheduling results with warabandi approach

Item	Starting Date	Irrigation (days)	Interval (days)
S1-T1	14th June	1	7
S1-T2	14th June	1	7
S1-T3	12th June	1	7
S1-T4	12th June	1	7
S2-T1	15th June	2	6
S2-T2	5th June	1	7
S2-T3	15th June	1	7
S2-T4	5th June	1	7
S3-T1	15th June	3	5
S3-T2	12th June	3	6
S3-T3	13th June	2	6
S3-T4	15th June	3	6
S4-T1	11th June	1	7
S4-T2	10th June	1	7
S4-T3	3rd June	1	7
S4-T4	10th June	1	7

7.5 Comparison on GA Formulations

The GA was able to produce feasible schedules under both the Zero-1 and Warabandi approaches. The Zero-1 approach is, however, inherently more flexible and resulted in both better equity and better overall water supply.

The Warabandi approach had a chromosome length of 48, compared to a chromosome length of 1600 for the Zero-1 approach. It therefore executes more efficiently but this does not improve its ability to meet demands when they are needed. The approach results in more water being supplied than is required in the early part of the schedule.

Figure 7.13 shows the total irrigation water supplies. Average cumulative water stress over the entire system is summarised in Figure 7.14. From these the advantages of the Zero-1 approach are clear.

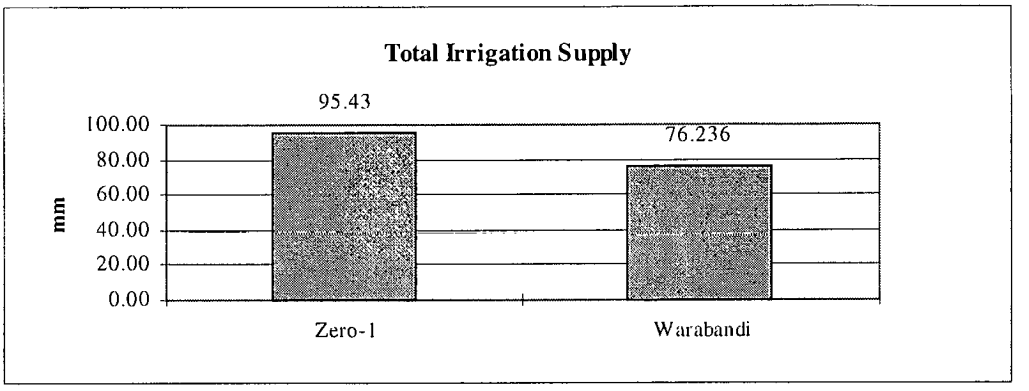

Figure 7.13 Total irrigation supply

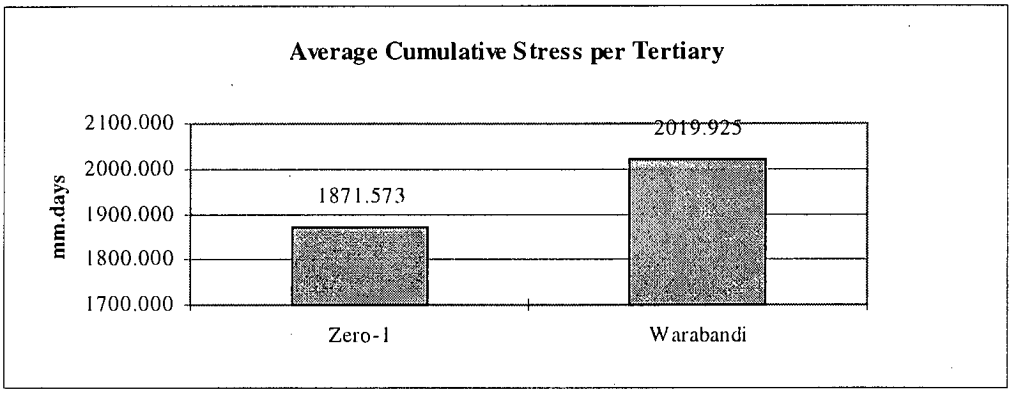

Figure 7.14 Average cumulative soil moisture stress

7.6 Evaluation of GA Operators and Sensitivity

Sensitivity analyses were carried out in much the same as reported in earlier chapters in order to determine whether changes in parameter levels, or in the GA operators used had any more influence on a more complicated network. These sensitivity analyses have been carried out only on the GA based on the zero-1 approach. As binary representation has been used, the conventional mutation operator was applied.

The sensitivity to crossover probability and to mutation probability has been assessed using tournament selection and uniform mutation. Figure 7.15 shows the sensitivity to crossover probability. Good results are achieved over a wider range of crossover probabilities, but the best results were clearly achieved with a crossover probability of 0.9. Sensitivity to mutation probability is shown in Figure 7.16. It was found that 0.09 mutations per chromosome, produced the best fitness.

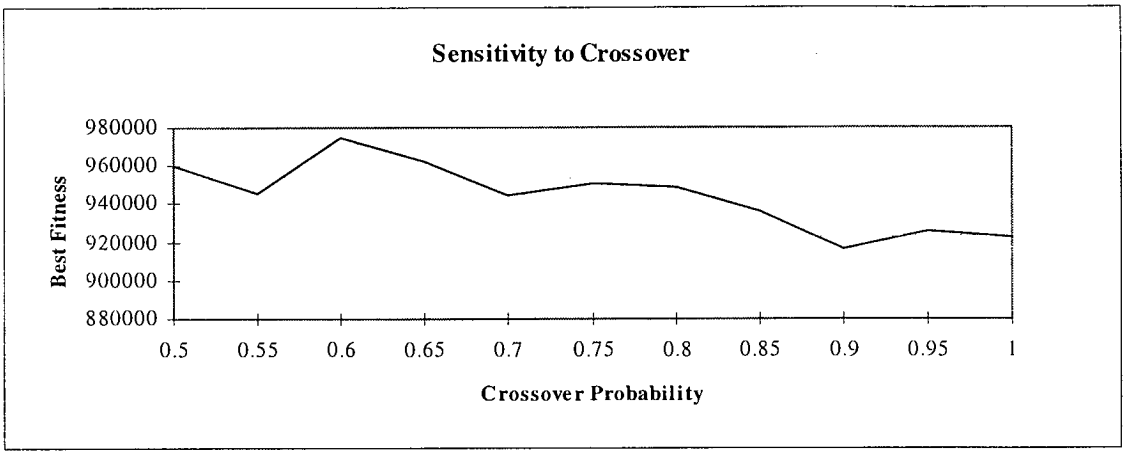

Figure 7.15 Sensitivity to crossover

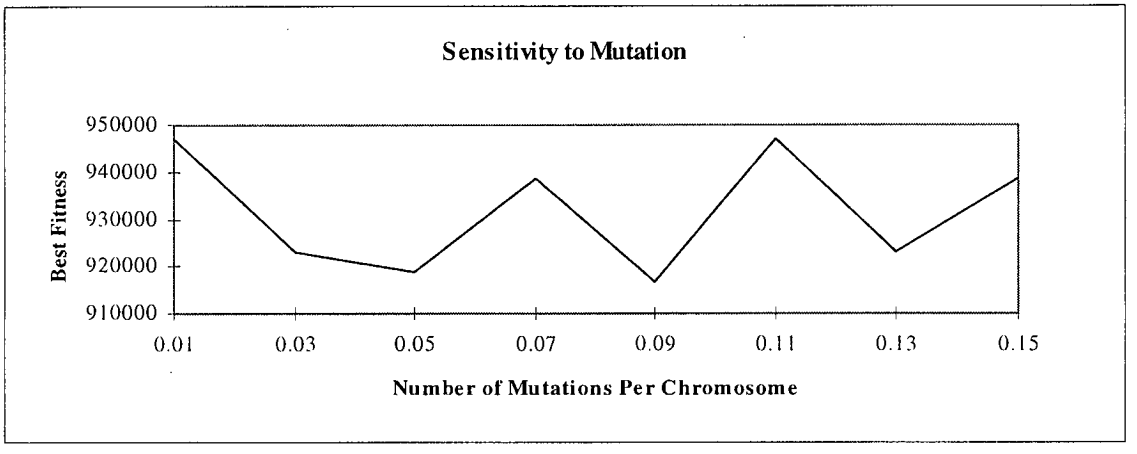

Figure 7.16 Sensitivity to mutation

8. APPLICATION OF GA TO WATER SCHEDULING IN PUGAL SYSTEM

8.1 Introduction

The Zero-1 and Warabandi GA approaches to water scheduling have been applied to part of the Indira Gandhi Nahar Pariyojana (IGNP) irrigation system in northwest India. Currently the warabandi approach to scheduling is used, and the opportunity existed to compare the warabandi approach with the alternative zero-1 approach through mathematical modelling. In fact scheduling is only normally carried out at minor and sub-minor canal level, but in scenarios considered here, scheduling has been investigated at distributary canal level.

The IGNP project was conceived in the late 1940's. Construction work began in 1958, and since then the project has continued to evolve. Figure 8.1 shows the current planned extent of the project, and Table 8.1 summarises the present status of development.

Table 8.1 Status of IGNP development in 1997

	Position in 1997		**Target**	
	Stage I	**Stage II**	**Stage I**	**Stage II**
Canal construction (km):				
Main canal	393	256	393	256
Distribution system	3,190	2,740	3,400	5,780
Area developed for irrigation (ha)	534,000	561,000	553,000	1,316,000
Area irrigated (2 seasons / year, ha)	670,000	150,000	553,000	964,000
Area served by lined water course (ha)	450,000	341,000	553,000	1,316,000

The project is on a very large scale, and improving water management is considered to be essential if productivity is to be maintained. In some areas waterlogging is a problem, and

salinity is becoming a problem. Improved water management would help overcome these problems. Another issue is that in order to extend the cultivable area in Phase II, water deliveries are to be reduced in the Phase I area. The difficulties associated with this are outlined in section 8.3.

For the purposes of evaluating GA performance, and the performance of the zero-1 approach against that of the warabandi approach, a branch canal system has been studied. Data were obtained from Mott MacDonald (1999) for the Pugal Branch Canal, which has a command area of about 50,000 ha.

8.2 The Pugal Branch Canal System

The Pugal Branch Canal has a length of 66 km, and irrigates an area of 49,394 ha. It has a capacity of 20.4 m^3/s. A schematic of the branch canal system is shown in Figure 8.1. There is a tree structure comprising distributary canals, minor canals and sub-minor canals. Table 8.2 summarises the characteristics of these canals. There are 5 distributary canals, 20 minor canals, and 4 sub-minor canals. The total length of distributary and minor canals is about 305 km.

The Pugal main canal and 30.5 km of distributary and minor canals are lined. For lined canals seepage losses are assumed to be 0.6 m^3/second/million m^2. In unlined canals seepage losses are taken to be 2.44 m^3/second/million m^2 (Mott MacDonald 1998). Mott MacDonald (1999) has translated these loss rates into conveyance efficiencies of 90% for lined canals, and 83% for unlined canals. The supply-based system has no drainage return flows to the canal system, although a separate drainage system may be required in some areas. From a modelling point of view, the canal system is much simpler than that associated with the Tukad Ayung system and discussed in Chapter 5.

Mott MacDonald (1998) has estimated field efficiencies to be of the order of 60%. Soil are predominantly loamy sand and sandy loam. In the Pugal system, average FC has been estimated to be 0.134 m^3/m^3 and average PWP to be 0.06 m^3/m^3. Double cropping is practised over part of the area. In the winter, mustard, gram and wheat area grown, while in the summer the crops are predominantly cotton, pulses and groundnut.

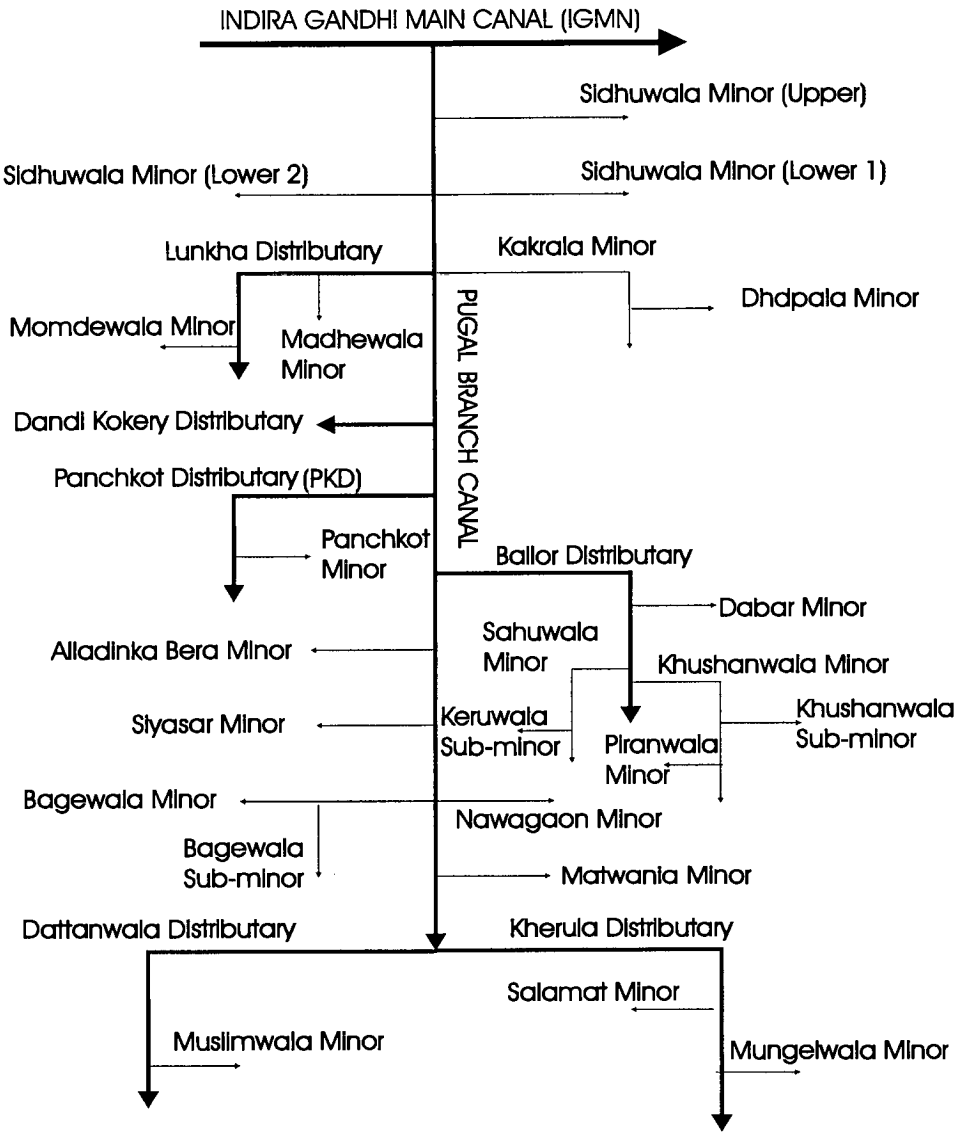

Figure. 8.1 Schematic of the Pugal branch canal

Table 8.2. Characteristics of the Pugal branch canal system

Canal Name	Canal Length (km)	CCA (ha)	Design Discharge (m3/s)
1 PUGAL BRANCH	66.02	49394	20.368
1.1 Sidhuwala Minor (Upper)	4.08	892	0.357
1.2 Sidhuwala Minor (Lower 1)	1.4	126	0.049
1.3 Sidhuwala Minor (Lower 2)	0.76	184	0.071
1.4 Kakrala Minor	12.77	1684	0.676
1.4.1 Dholpala Minor	3.2	426	0.167
1.5 Lunkha Disty	28.68	5050	2.013
1.5.1 Madhewala Minor	5.18	641	0.254
1.5.2 Momdewala Minor	3.05	505	0.197
1.6 Dandikokery Disty	16.92	3270	1.272
1.7 Panchkot Disty	22.15	5584	2.264
1.8 Ballar Disty	48.46	12720	5.247
1.8.1 Dabar Minor	1.83	422	0.163
1.8.2 Sahuwala Minor	5.7	949	0.373
1.8.2.1 Kheruwala Sub-minor	0.94	382	0.147
1.8.3 Khushanwala Minor	32.61	5859	2.353
1.8.3.1 Khushanwala Sub-minor	1.46	298	0.116
1.8.3.2 Peranwala Minor	7.5	1385	0.549
1.9 Alladinka bera Minor	3.89	685	0.327
1.10 Siyasar Minor	12.44	2286	0.896
1.11 Nawafgoan Minor	5.94	378	0.148
1.12 Matwania Minor	6.93	565	0.224
1.13 Bagewala Minor	4.26	967	0.395
1.13.1 Bagewala Sub-minor	0.91	253	0.098
1.14 Kherulla Disty	38.4	5390	2.235
1.14.1 Salamat Minor	1.95	418	0.162
1.14.2 Mungelwala Minor	5.64	434	0.173
1.15 Dattanwala Disty	22.55	5123	2.016
1.15.1 Muslimwala Minor+A12	5.79	1083	0.416

8.3 Present Operational Practise in the IGNP Project

The IGNP system is operated by the warabandi approach. At watercourse level, each land holder receives the entire discharge of the watercourse in turn, with the duration of each turn being determined in proportion to the size of the land holding. In the IGNP, the duration of the warabandi rotation is one week (168 hours), and farmers get their allocation at a predefined time each week throughout the season. Superimposed on this is a system of rotational running of canals. Canals must either run at their full design discharge, or be empty. The reasons for this are that watercourses are generally ungated, and a higher or lower level in the canal would result in an inequitable distribution of water between watercourses; and that the canals have been designed as regime canals with the intention that silt should pass through the system and onto farmers' fields – lower than design discharge results in sedimentation.

The rotational period for canal operation may vary as a result of variations in water availability, and as a result of variations in water requirements. Current practise is to rotate in multiples of 7 days. The framework and constraints for present operation are summarised below:

a) A constant supply is maintained at the head of the main canal for pre-defined periods, which can vary from a minimum of 10 days to more than 30 days. Flows in the main canal will generally be the lower of the flow required to meet design demands in the system, or the maximum available flow constrained by river flow.

b) Branch, distributary, minor and sub-minor canals are operated under full capacity except during filling and following closure of the branch offtake.

c) The irrigation interval for any farmer should not exceed the critical period associated with the stage of crop growth and root zone development as determined according to design cropping patterns.

d) The preferred period for water distribution is 7 days, and if this is varied, account should be take of losses associated with canal filling and emptying.

8.4 Modelling Scheduling for the Pugal System

Scheduling in the Pugal System has been modelled following the approach outlined in Chapters 6 and 7. The fundamental objective is that of equation 6.3. For evaluation purposes, it has been assumed that the entire area is cropped with cotton as it is the most dominant crop grown in summer, with preferred planting between the 1st and 15th April, and a total growing period of 195 days. Computed crop water requirements are summarised in Table 8.3 below.

Table 8.3 Crop water requirements (mm/month)

Month	Apr	May	Jun	Jul	Aug	Sep	Oct	Total
Cotton	82.5	268.1	123.8	123.8	123.8	123.8	61.9	907.5

For cotton, the minimum rooting depth was taken to be 0.3 m, and the maximum rooting depth 1.4 m. A soil water depletion fraction of 0.65 was assumed (Allen et. al. 1998).

Modelling has been carried out of canals fed directly from the Pugal Branch canal, and each of these is in effect treated as a scheme. The objective has been to determine an appropriate rotational schedule for operating of these canals. There are 15 canals fed directly from the Pugal Branch canal. The canals would normally be operated in groups such that the capacity of the groups closely matched the (capacity) discharge of the branch canal. The GA can, however, optimise the starting time for each canal, and form its own groups. In the Pugal Branch canal, the sum of the capacities of the distributaries is almost equal to the capacity of the Branch canal, and if the Branch canal were running full, grouping and rotation between distributaries would not be required. However, the condition investigated is one of water stress in which the Branch canal is not running full.

Two GA approaches have been set up. A warabandi approach has been set up with a warabandi period of 7 days. The zero-1 approach was also set up. Canal efficiency have been taken to be 90%. Table 8.4 summarises the additional losses associated with canal opening and closure.

Table 8.4 Canals in operation and scheduling by GA

Canal No.	Canal Names	CCA (ha)	Design Discharge (m^3/sec)	Losses per Irrigation (mm)
1	Sidhuwala Minor (Upper)	892	0.344	1.160
2	Sidhuwala Minor (Lower 1)	126	0.049	0.401
3	Sidhuwala Minor (Lower 2)	184	0.072	0.219
4	Kakrala Minor	1684	0.676	3.778
5	Lunkha Disty	5050	2.014	8.429
6	Dandikokery Disty	3270	1.272	4.850
7	Panchkot Disty	5584	2.265	6.621
8	Ballar Disty	12720	5.249	14.737
9	Alladin ka bera Minor	685	0.328	1.373
10	Siyasar Minor	2286	0.897	3.597
11	Nawagoan Minor	378	0.148	1.714
12	Matwania Minor	565	0.224	2.025
13	Bagewala Minor	967	0.395	1.282
14	Kherulla Disty	5390	2.236	11.739
15	Dattanwala Disty	5123	2.018	6.546

The models were set up to operate with the branch canal operating at full capacity, and at 75% and 50% of full capacity. Although the branch canal was permitted to run at less than full capacity, distributaries and minors always had to run at their full capacity.

Both GAs were set up with tournament selection, conventional mutation and uniform crossover. A population size of 100 was chosen for all runs, and a crossover probability of 0.9 used. The number of mutations per chromosome was set to 0.09. With 195 time steps being modelled, 15 schemes, and a gene representing starting time period in each scheme, the total chromosome length was 2940 for the zero-1 approach, which is significantly longer than in any of the test networks used. For the warabandi approach, the total chromosome length was 45. The maximum number of generations allowed was set to 2500, but the GAs were considered to have converged if the improvement in the fitness function between any 300 generations was less than 0.001%.

8.5 Evaluation of Results

The irrigation schedules produced by the zero-1 approach are presented in Figures 8.2 to 8.4, and by the warabandi approach are presented in Figures 8.5 to 8.7 for each of the branch canal discharges assumed. In the warabandi approach, rotation is assumed at the flexible increments in the range of 1 to 7 days. It was found that canal capacities constraints were satisfied in all canal discharges assumed. The water supply to the system when Pugal canal supplies at full supply, 75% supply and 50% were 502 mm, 301 mm and 60 mm by zero-1 approach, and 318 mm, 231 mm and 72 mm by warabandi method. All cases were in the water stress condition. When the Pugal canal is supplied at full capacity it is possible for all 15 canals to irrigate simultaneously. When the Pugal canal supplied at 75% and 50% of full capacity, scheduling was required. These cases resulted in severe water stress. It is of interest to note that canal opening and closing losses are significant, and the GA adopts a strategy that minimise these.

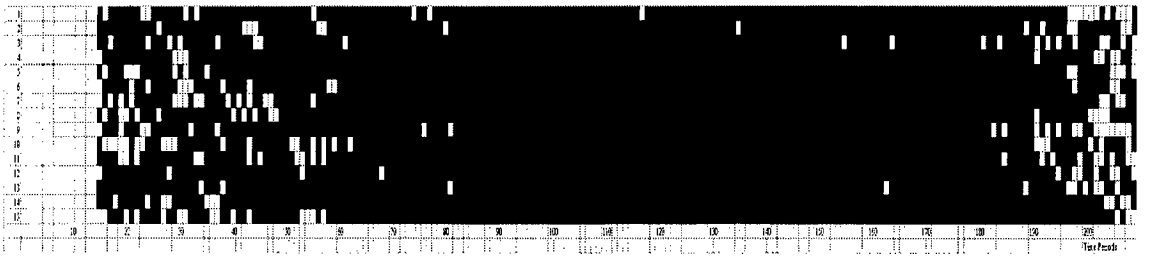

Figure 8.2 Irrigation schedule generated by zero-1 approach, Pugal full supply

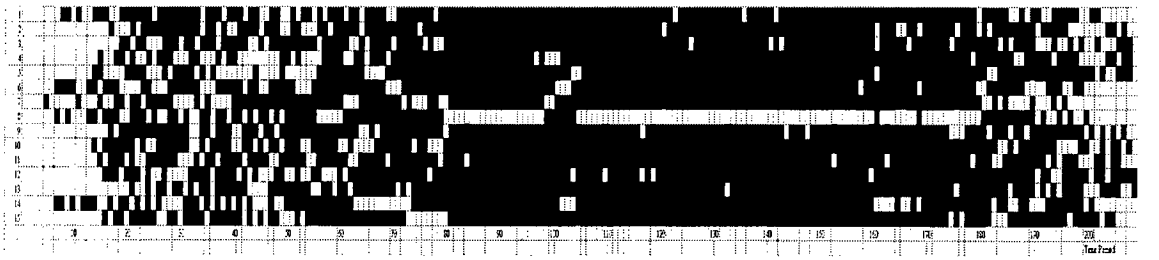

Figure 8.3 Irrigation schedule generated by zero-1 approach, Pugal supplies 75%

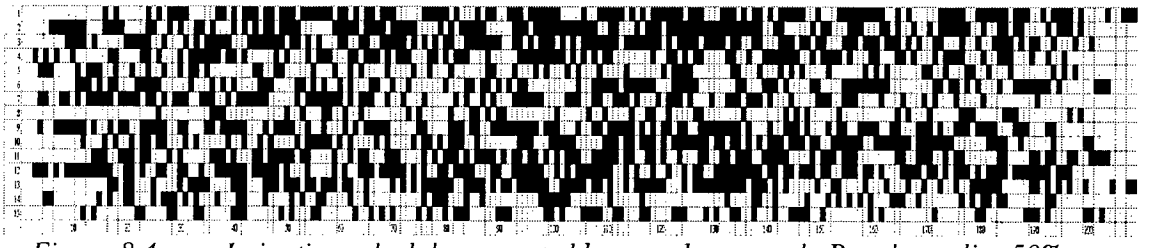

Figure 8.4 Irrigation schedule generated by zero-1 approach, Pugal supplies 50%

Figure 8.3 and 8.4 shows that canal number 8 (Ballar distributary) is heavily deprived of water in times of water scarcity. This canal supplies the most largest area (12720 ha), and

the losses per gate opening and shutting is highest among other canals. Once the canal is operated, it takes at least 5 days in delivering water to the planting area. If this canal is chose to operate, the system canals capacity constraint will be very high. GA automatically let this canal become most stress in the stress period. However, the real situation is that the canal system is operated by full supply in distributary and minor canals.

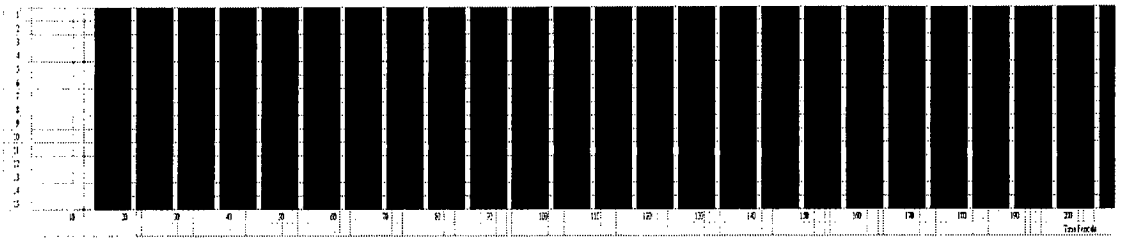

Figure 8.5 Irrigation schedule generated by warabandi, Pugal full supply

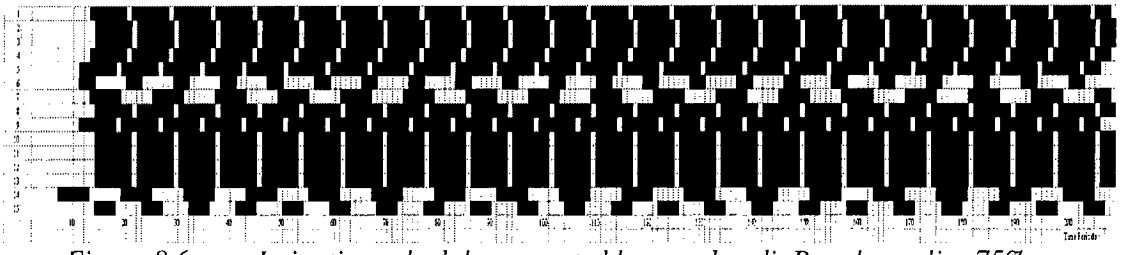

Figure 8.6 Irrigation schedule generated by warabandi, Pugal supplies 75%

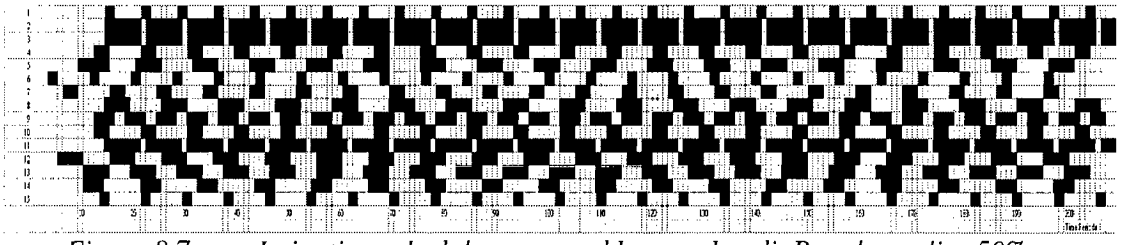

Figure 8.7 Irrigation schedule generated by warabandi, Pugal supplies 50%

A summary of corresponding water stress conditions is presented in Figures 8.8 to 8.10 for zero-1 approach, and Figures 8.11 to 8.13 for the warabandi approach. These figures present in Ea/ETc ratio for each scheme under the range of canal discharges and for each scheduling approach. It has been found that when Pugal supplied at full capacity the Ea/ETc ratio were in the range of 0.459 to 0.625 with zero-1 approach, and 0.233 to 0.560 with warabandi approach. Significant stress thus exists in the system even without any additional water shortage being introduced. Any significant reduction in canal flows would almost certainly result total crop failure. The Ea/ETc ratio when the Pugal supplied at 75%, were in the range of 0.132 to 0.566 with zero-1 approach, and. 0.083 0.493 with warabandi approach. It is clear that the scheduling with the zero-1 approach produces a more equitable distribution of water stress than the warabandi method.

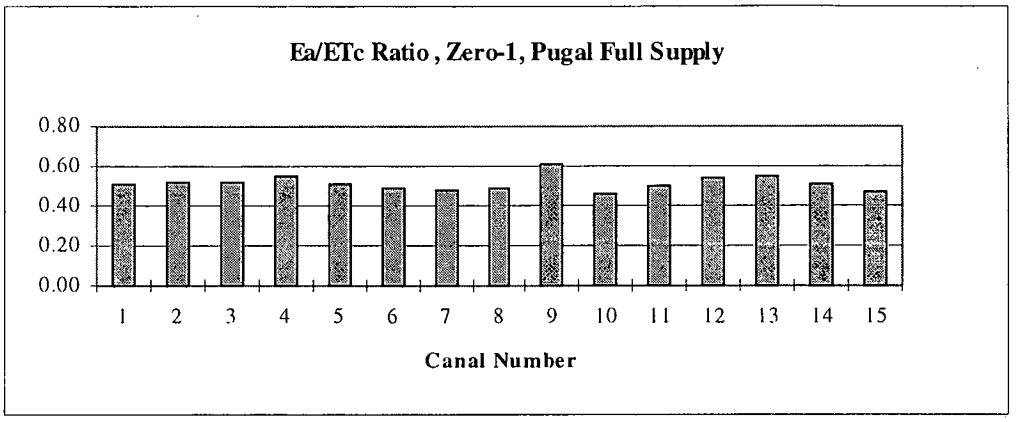

Figure 8.8 Ea/ETc ratio with zero-1 approach, Pugal full supply

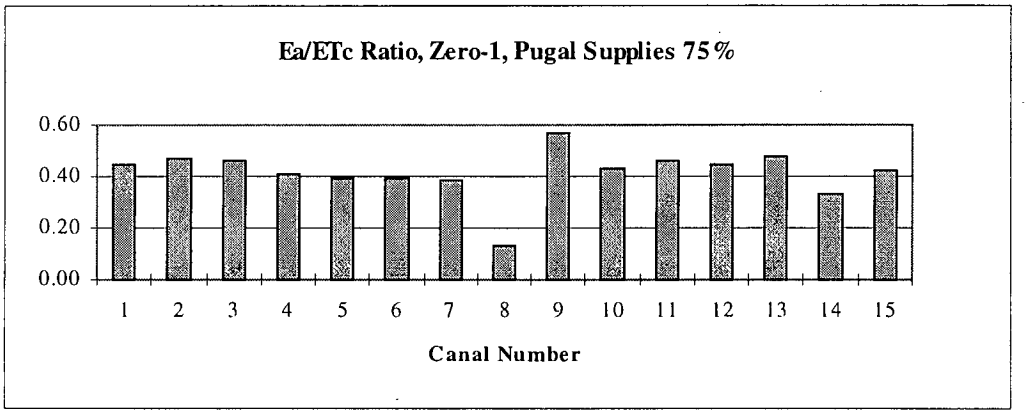

Figure 8.9 Ea/ETc ratio with zero-1 approach, Pugal supplies 75%

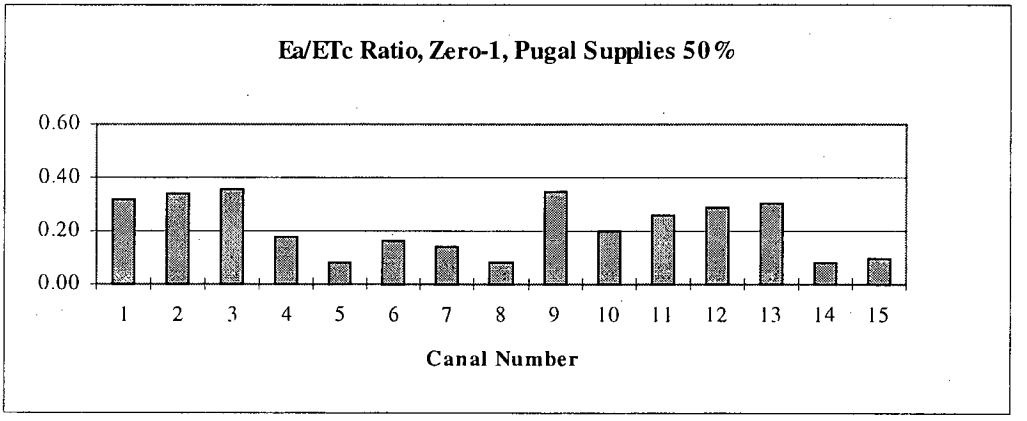

Figure 8.10 Ea/ETc ratio with zero-1 approach, Pugal supplies 50%

Cumulative soil moisture stress were also determined in mm.days for each of assumed canal discharges and approaches. With Zero-1 approach when Pugal canal at full supply, 75% and 50% the average cumulative stresses were 3365 mm, 4200 mm, and 5829 mm. With the warabandi approach the corresponding figures were 4059 mm, 4451 mm, and 5495 mm respectively.

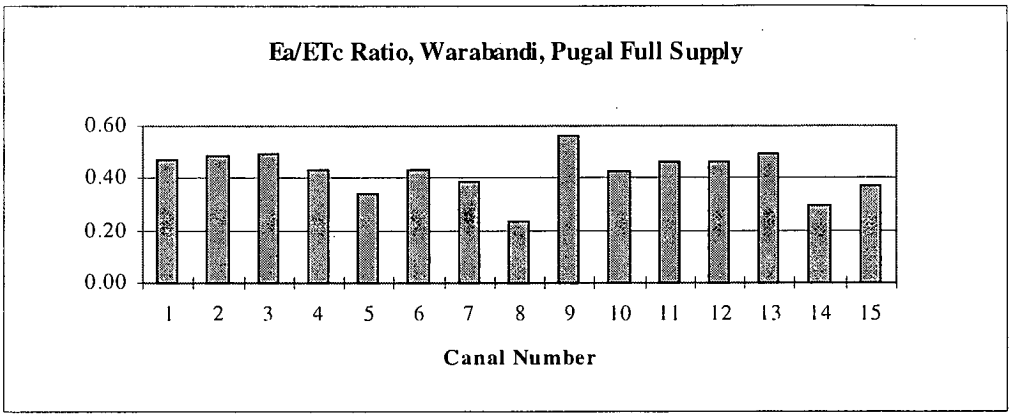

Figure 8.11 Ea/ETc ratio with warabandi approach, Pugal full supply

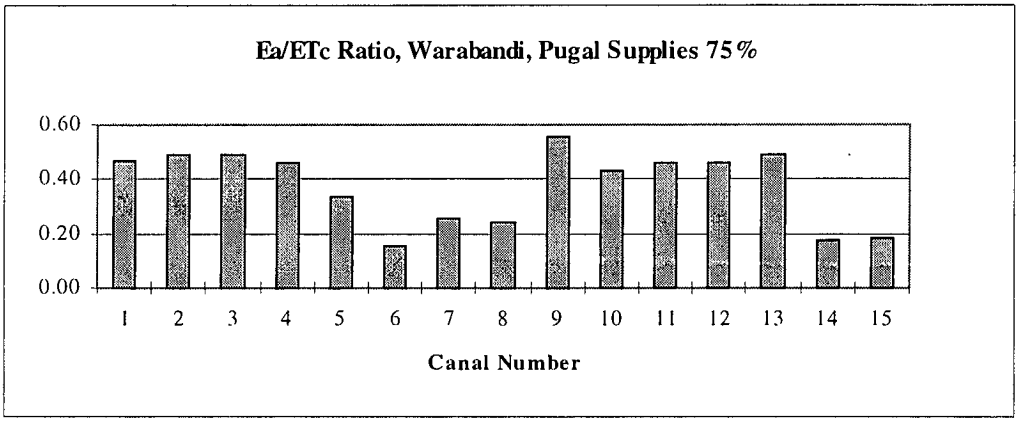

Figure 8.12 Ea/ETc ratio with warabandi approach, Pugal supplies 75 %

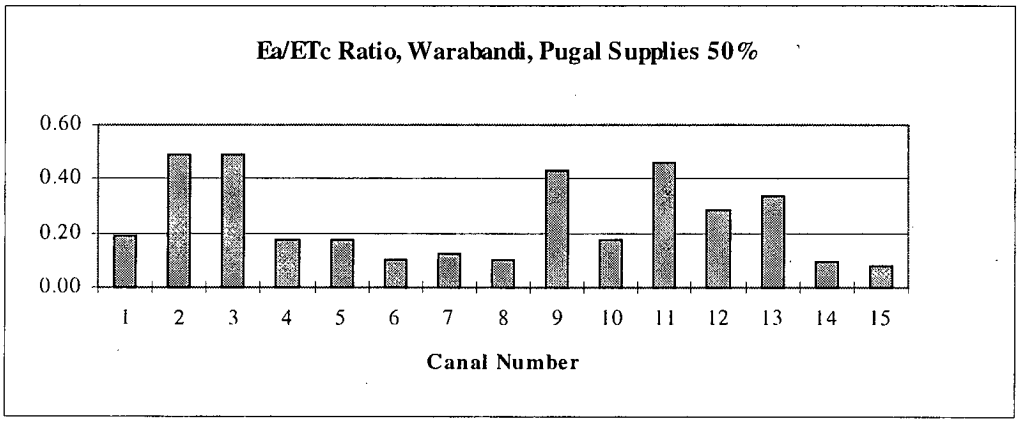

Figure 8.13 Ea/ETc ratio with warabandi approach, Pugal supplies 50%

Soil moisture variations are presented for two representative schemes in Figures 8.14 to 8.16 for zero-1 approach, and Figure 8.17 to 8.19 for warabandi approach for each of the assumed main canal discharges. As stated before, even under full supply there is significant water stress, and soil moisture content is below wilting point throughout most of the development stage of the crop.

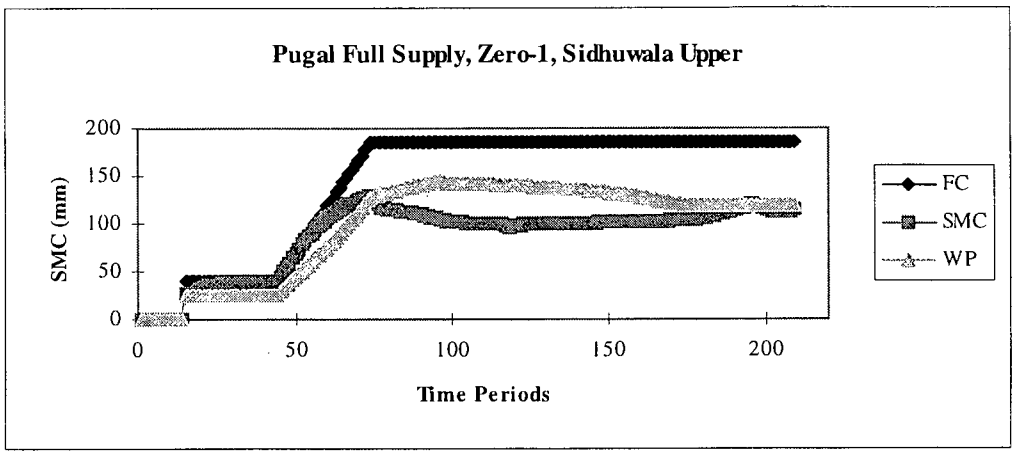

Figure 8.14 Simulated root zone soil moisture content with zero-1 approach (Pugal full supply)

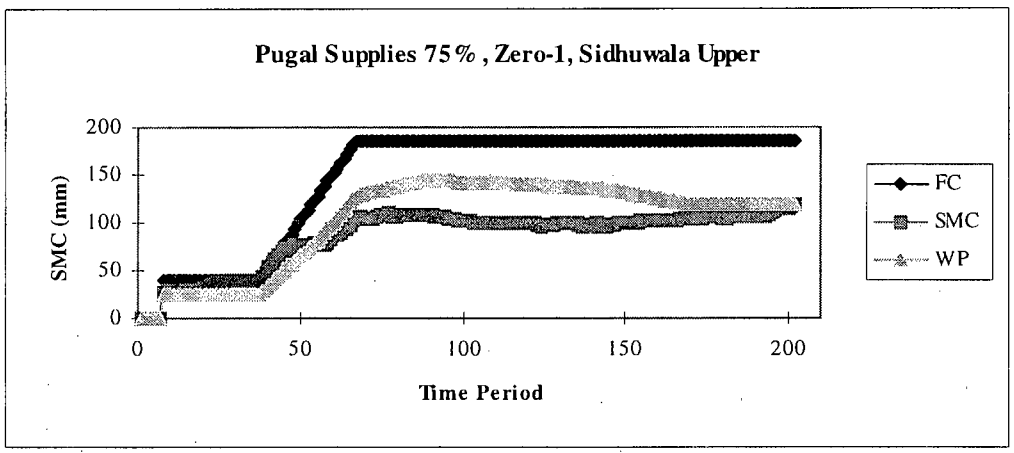

Figure 8.15 Simulated root zone soil moisture content with zero-1 approach (Pugal supplies 75%)

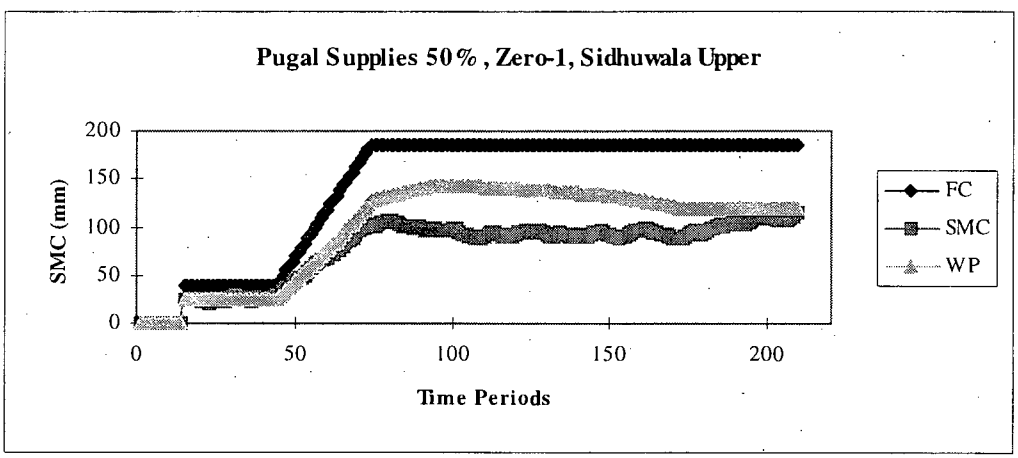

Figure 8.16 Simulated root zone soil moisture content with zero-1 approach (Pugal supplies 50%)

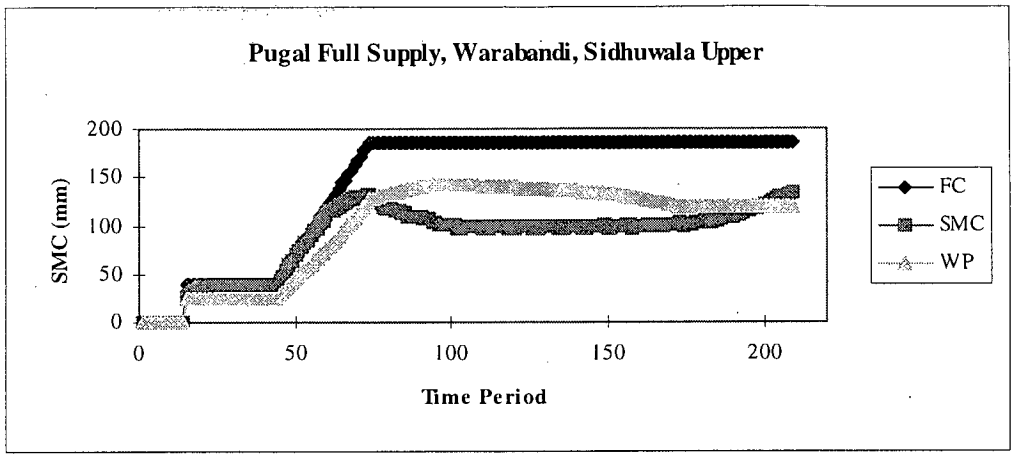

Figure 8.17 Simulated root zone soil moisture content with warabandi approach (Pugal full supply)

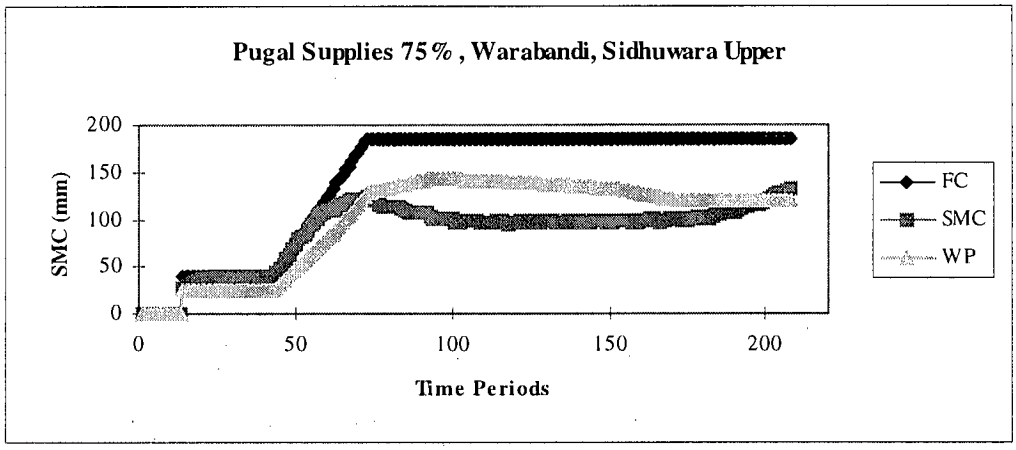

Figure 8.18 Simulated root zone soil moisture content with warabandi approach (Pugal supplies 75%)

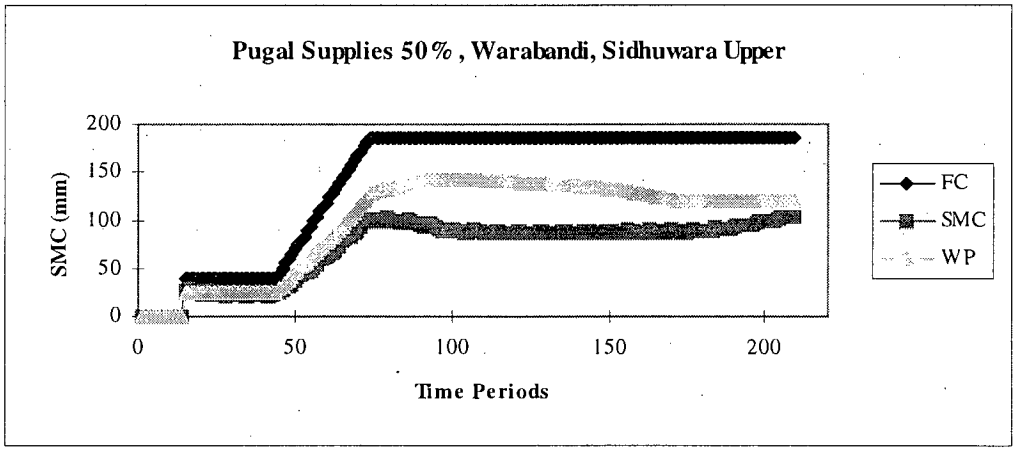

Figure 8.19 Simulated root zone soil moisture content (Pugal supplies 50%)

Table 8.5 summarises the planting dates for each canal. Under the warabandi system it shows the irrigation days and also the minimum interval period between irrigations. Optimal planting dates under full supply do not vary between schemes. However, with reduced supply and enforced scheduling between schemes, a wider range of planting dates is required. The GA handles this very well under both formulations.

Table 8.5 Summation of start of planting date, irrigation days and interval

Canal No.	Zero-1			Warabandi								
	Full	75%	50%	Full			75%			50%		
	start	start	start	start	Irrig. (days)	Intvl. (days	start	Irrig. (days)	Intvl (days)	start	Irrig. (days)	Intvl. (days)
1	15th	7th	15th	15th	7	1	14th	7	1	15th	2	5
2	14th	14th	5th	15th	7	1	15th	7	1	15th	7	1
3	14th	14th	9th	15th	7	1	15th	7	1	15th	7	1
4	15th	13th	3rd	15th	7	1	14th	7	1	13th	3	6
5	15th	13th	4th	15th	7	1	12th	7	1	10th	5	6
6	15th	5th	11th	15th	7	1	12th	3	6	4th	2	6
7	15th	2nd	4th	15th	7	1	14th	6	6	7th	3	6
8	15th	6th	9th	15th	7	1	15th	7	1	15th	5	6
9	15th	14th	1st	15th	7	1	12th	7	1	13th	4	2
10	15th	13th	8th	15th	7	1	15th	7	1	13th	3	6
11	14th	14th	11th	15th	7	1	15th	7	1	15th	7	1
12	14th	14th	3rd	15th	7	1	15th	7	1	6th	5	5
13	15th	14th	15th	15th	7	1	15th	7	1	11th	5	4
14	15th	6th	2nd	15th	7	1	8th	6	6	11th	4	7
15	15th	13th	10th	15th	7	1	15th	4	5	14th	2	6

8.6 Sensitivity Analyses

Sensitivity analysis was carried out on the zero-1 GA, and yielded very similar results to those obtained on the test networks. The sensitivity to crossover probability is shown in Figure 8.20. Clearly reasonable results were obtained of a wide range of crossover

probability, but a value of 0.9 yielded the best results. This compares with a value of 0.85 for the test network in Chapter 6 and 0.9 in Chapter 7.

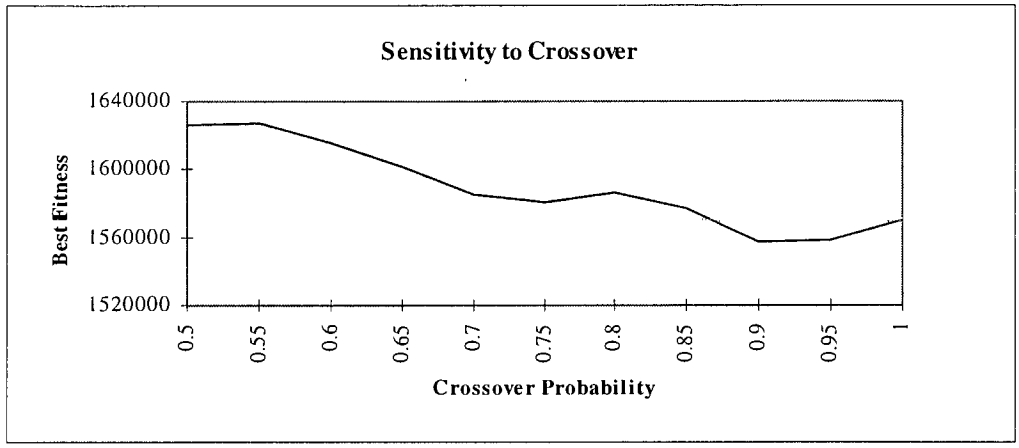

Figure 8.20 Sensitivity to crossover

Sensitivity to mutation probability was assessed with between 0.03 and 0.15 mutations per chromosome. The results are presented in Figure 8.21. Clearly at higher mutation probabilities, it is difficult to maintain good solutions, and a value of 0.09 mutations per chromosome yielded the best results. There would therefore in a population of 100, only be a mutation in one chromosome per generation on average. This is consistent with the results presented in chapter 7.

All cases were satisfied capacity constraint when using penalty factor 1 for R1 and 10000 for R2.

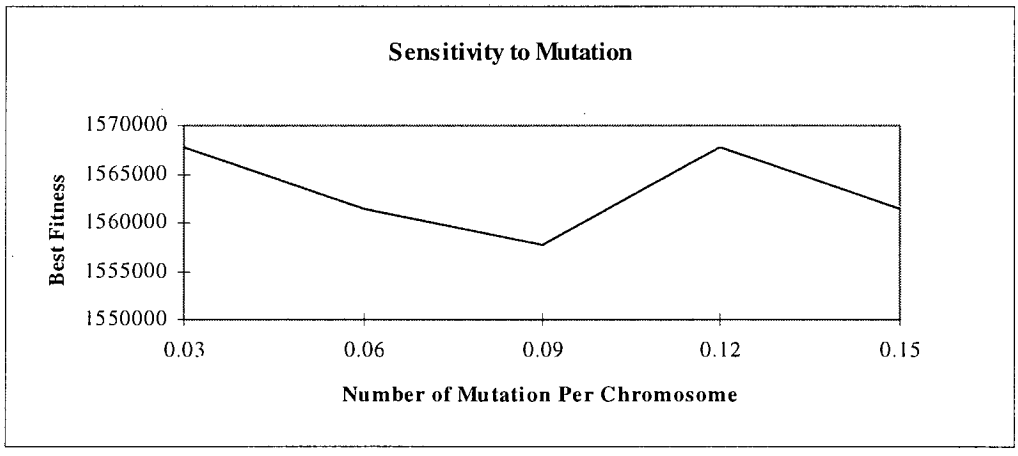

Figure 8.21 Sensitivity to mutation

8.7 Conclusions

It has been demonstrated that the GA set up with the zero-1 approach produced a more equitable irrigation schedule than did the warabandi approach. Zero-1 approach is more flexible in rotational irrigation and responds to root zone soil moisture crop water requirements. The traditional warabandi approach cannot do this. The GAs have been shown to be flexible in application to this type of problem.

9. APPLICATION OF GA TO A LATERAL CANAL SCHEDULING PROBLEM

9.1 Introduction

The water scheduling problem considered in Chapters 6 to 8 was controlled by root zone soil moisture requirements, and calculation of irrigation requirements was implicit in the approach. In this Chapter, the application is to a system in which water requirements are considered to be known, and the objective is to determine the timing of allocations to schemes that maximises water use efficiency. The water scheduling problem addressed was the subject of recent studies by Reddy et al (1999).

Reddy et. al (1999) have described the development of a 0-1 linear programming approach to the solution of a lateral canal scheduling problem. The problem addressed was one in which irrigation supplies to a large number of laterals, with different capacities and required running times, were to be scheduled. The requirement was that each lateral canal received its share of water during a specified rotational period. Reddy et. al. (1999) formulated the problem in terms of a required main canal capacity. The required main canal capacity is the sum of the lateral canal capacities running concurrently, and this should not exceed the actual canal capacity. The objective was to minimise the differences between the actual and the required main canal capacity. In their paper, Reddy et. al. (1999) presented an application of their approach to the Hetao irrigation project in China. The data were in sufficient detail to permit a GA approach to the problem to be set up, and results compared with those of the 0-1 linear programming approach adopted by Reddy et. al..

9.2 The 0-1 LP Formulation

In the 0-1 LP formulation developed by Reddy et. al., the rotation period could be subdivided into quarter, half or full day time blocks. The decision variable was X_{ij}, which could take a value of 1 if lateral j started in time step i, but was otherwise set to 0. The objective was to minimise the penalties associated with main canal capacity being exceeded. The formulation of the penalty function was not well explained, and was of the form:

$$\text{Minimise } Z = \sum_{k=1}^{N_{level}} C_k Y_k \qquad (9.1)$$

where;

C_k = penalty for exceeding the supply canal capacity by level kth, ($)

Y_k = kth level of increase in flow rate capacity of the supply canal

N_{level} = the number of levels of flow rate increases considered

No indication was given of how Y_k was established, although it is assumed that it must have been from the canal capacity constraint given in equation 9.2 below.

$$\sum_{j=1}^{N} q_j \sum_{m=1}^{T+1-\delta_j} \alpha_{imj} X_{mj} - \sum_{k=1}^{N_{level}} X_k \le Q \qquad \text{for each } i \qquad (9.2)$$

where;

i = time step

q_j = capacity of lateral j

Q = capacity of the supply canal

m_j = the starting time period index for secondary j

$\propto_{imj}$ = contribution of secondary j that started operating during the mth time period, to the flow rate in the canal during the ith time period. This is stated mathematically as:

$$\propto_{imj} = 1 \quad \text{if} \quad m_j \le i \le m_j + \delta_j - 1 \qquad (9.3)$$

The meaning of N_{level} is not clear, nor is the logic by which X is expressed with different sub-scripts in equation 9.2. It is assumed that the confusion may arise through some level of grouping offtakes.

9.3 Formulation by GA

The formulation used for the GA is somewhat more transparent than that described by Reddy et. al. (1999). The objective is to minimise the differences between main canal capacity and the sum of the lateral canal capacities operating in any particular time step:

$$\text{Minimise } Z = \sum_{i=0}^{n}(Q_i - \sum_{j=1}^{m}(q_{ij}.IFLAG_{ij}))^2 \qquad (9.4)$$

where;

Q_i = supply canal capacity (m^3/s)

q_{ij} = capacity of lateral j (m^3/s)

$IFLAG_{ij}$ = indicator for lateral operation, value of 0 or 1

m = number of laterals

n = number of time periods in the rotation

The value of *IFLAG* is determined from the starting time step of lateral j. The operating time for each lateral is known, and the starting time step $ISTRT_j$ is used to define the value of $IFLAG_{ij}$.

$$IFLAG_{ij} = 0$$

$$If \quad i \geq ISTRT_j \text{ and } i < (ISTRT_j + OT_j), \quad IFLAG_{ij} = 1 \qquad (9.5)$$

where, OT_j is the operating time for lateral j. The decision variable for the problem is *ISTRT*. Expressing the objective function in quadratic form ensures that differences of opposite sign do not cancel each other out. Additionally a penalty function is introduced to ensure that infeasible solutions in which the main canal capacity is exceeded are not produced. The penalty function is expressed as follows:

$$If \sum_{j=1}^{m} q_{ij} > Q_i \quad P_1 = \sum_{i=1}^{n}\left(\sum_{j=1}^{m} q_{ij} - Q_i\right) \qquad (9.6)$$

P_1* is added to the objective function.

*In this system, a linear form of the penalty function proved adequate. In a more complex system it is expected that a quadratic form would be required.

9.4 Application to the Hetao Irrigation Project

9.4.1 General

Reddy et. al. (1999) demonstrated the application of their 0-1 LP approach on the Xi Le Submain of the Hetao irrigation project, which is situated in the Inner Mongolia Region of China. The supply canal has a capacity of 22.6 m^3/s and feeds 117 lateral canals. The later canals vary in capacity between 0.4 m^3/s and 3.0 m^3/s. The data for the system published by Reddy et. al. are presented in Table 9.1. The column indicating time blocks required is for 12 hour time periods. In their application, Reddy et. al. (1999) specified the time periods within which irrigation from each lateral should begin. This grouping was intended to represent common practice, and has been defined in a way that delivers water firstly to laterals at the head of the system, and lastly to the those at the tail. The optimal starting times obtained by Reddy et. al. are included in Table 9.1.

In setting up the GA for the system, two approaches have been considered, one in which the starting time periods were pre-specified, as assumed by Reddy et. al., and one in which the GA had complete freedom to select the most efficient lateral starting times. In each case, the chromosome length was 117. It has been found that the GA approaches converge very quickly to a solution, and can do so with a relatively small population. It was found that a population of 8 was sufficient to reach a solution. A solution was deemed to have been found when the improvement in fitness over 300 generations was less than 0.001%.

9.5 Evaluation of Results

Table 9.2 presents a comparison of the optimised starting time blocks when these were to be within the pre-specified range given in Table 9.1, and when the GA was given the freedom to select the most appropriate starting time block. When the GA was run with the constraint of starting time blocks being within a pre-specified range, results were similar to those presented in Table 9.1, but not exactly the same. The GA was able to satisfy all constraints, and this resulted in a slightly different, but more correct solution. The starting time blocks produced by each of the GA formulations are presented in Table 9.2. Permitting the GA freedom to select the most appropriate starting block for each lateral results in significantly different results, although it is recognised that hydraulic conditions may prevent practical application of the schedule produced.

Table 9.1 Data for Xi Le Submain, Hetao Irrigation Project

Secondary Number	Capacity m^3/s	Operation Time (h)	Volume per canal (m^3)	Time block required	Pre-specified starting time period	Result of Optimal starting time period
(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	0.4	50	72000	4	1 to 5	1
2	0.8	72	207360	6	1 to 5	1
3	0.4	49	70560	4	1 to 5	1
4	0.5	46	82800	4	1 to 5	1
5	0.8	74	213120	6	1 to 5	1
6	0.4	40	57600	3	1 to 5	1
7	1.0	85	306000	7	1 to 5	1
8	0.6	41	88560	3	1 to 5	1
9	0.4	32	46080	3	1 to 5	1
10	0.5	39	70200	3	1 to 5	1
11	0.6	67	144720	6	3 to 7	3
12	1.0	133	478800	11	2 to 6	2
13	0.4	58	83520	5	2 to 6	2
14	0.8	64	184320	5	2 to 6	2
15	0.7	57	143640	5	2 to 6	2
16	0.4	57	82080	5	2 to 6	2
17	0.4	42	60480	4	2 to 6	2
18	0.4	32	46080	3	2 to 6	2
19	0.6	81	174960	7	2 to 6	2
20	0.4	14	20160	1	2 to 6	2
21	0.4	66	95040	6	3 to 7	3
22	1.0	112	403200	9	3 to 7	3
:	:	:	:	:	:	:
:	:	:	:	:	:	:
:	:	:	:	:	:	:
:	:	:	:	:	:	:
114	0.6	63	136080	5	13 to 17	17
115	0.5	56	100800	5	13 to 17	17
116	0.3	6	6480	1	17 to 21	20
117	0.6	64	138240	5	13 to 17	17

Table 9.3 presents the summation of lateral flows in each time period for each of the three formulations. These results are also presented graphically in Figure 9.1. It is clear that GA with freely selected starting time blocks results in more uniform utilisation of the available main canal flow throughout the rotation cycle. The GA, even with pre-defined starting time block ranges, is clearly much more effective than the 0-1 LP approach developed by Reddy et. al., resulting in greater water delivery, and no constraint violations. The 0-1 LP approach has significant constraint violations between time steps 5 and 17. The maximum supply canal discharge is 22.6 m^3/s.

Table 9.2 Comparison Results of Pre-specified and Freedom Starting Time Blocks

Canal No. (1)	Pre-specified start range (2)	GA selected start (3)	Canal No. (1)	Pre-Specified start range (2)	GA selected start range (3)	Canal No. (1)	Pre-Specified start range (2)	GA selected start range (3)
1	1	21	**41**	8	21	**81**	18	20
2	1	14	**42**	9	19	**82**	19	9
3	1	14	**43**	9	14	**83**	17	15
4	1	3	**44**	5	8	**84**	17	6
5	1	19	**45**	5	1	**85**	13	20
6	1	9	**46**	7	8	**86**	18	9
7	1	12	**47**	9	6	**87**	16	3
8	1	15	**48**	7	14	**88**	16	1
9	1	17	**49**	9	6	**89**	17	2
10	1	22	**50**	7	6	**90**	16	8
11	3	19	**51**	6	4	**91**	17	17
12	2	12	**52**	7	8	**92**	17	19
13	2	3	**53**	6	20	**93**	20	10
14	2	9	**54**	8	12	**94**	17	1
15	2	4	**55**	6	1	**95**	17	19
16	2	16	**56**	9	22	**96**	17	2
17	2	1	**57**	10	6	**97**	19	12
18	2	4	**58**	7	3	**98**	18	15
19	2	8	**59**	9	19	**99**	17	4
20	2	1	**60**	11	4	**100**	18	13
21	3	19	**61**	10	9	**101**	2	2
22	7	16	**62**	11	1	**102**	14	14
23	3	4	**63**	10	19	**103**	16	8
24	3	8	**64**	7	21	**104**	11	16
25	3	15	**65**	11	13	**105**	12	19
26	3	11	**66**	8	15	**106**	11	11
27	3	6	**67**	7	21	**107**	13	6
28	3	15	**68**	9	7	**108**	17	9
29	8	7	**69**	11	21	**109**	13	3
30	4	2	**70**	12	9	**110**	17	10
31	4	9	**71**	11	13	**111**	16	12
32	4	4	**72**	12	20	**112**	19	22
33	4	20	**73**	12	4	**113**	17	13
34	5	9	**74**	12	19	**114**	17	1
35	4	10	**75**	11	16	**115**	17	20
36	4	20	**76**	11	1	**116**	21	16
37	4	3	**77**	10	17	**117**	17	7
38	4	17	**78**	12	19			
39	8	18	**79**	12	17			
40	5	17	**80**	16	11			

Table 9.3 Comparison on Summation of Lateral Flows by Formulations (m^3/s)

Time Period	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
LP-specified	8.8	13.9	19.7	23.5	25.4	25.1	25.4	25.3	25.4	25	22.7	21.6
GA-specified	5.8	13.9	18.7	21.1	21.4	21.3	21.8	21.6	21.7	21.2	21.5	21.9
GA-free	16.0	18.1	18.2	17.8	18.1	17.8	18.3	17.5	17.9	18.0	18.0	18.0
Time Period	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
LP-specified	21.0	23.3	24.9	25.4	24.9	18.9	11.5	7.2	5.2	3.0	3.0	0.0
GA-specified	20.3	20.2	18.8	20.4	22.3	21.8	20.9	19.4	18.6	9.5	3.0	3.0
GA-free	18.1	17.8	18.1	17.8	17.8	18.1	18.0	18.3	18.1	18.3	18.3	17.7

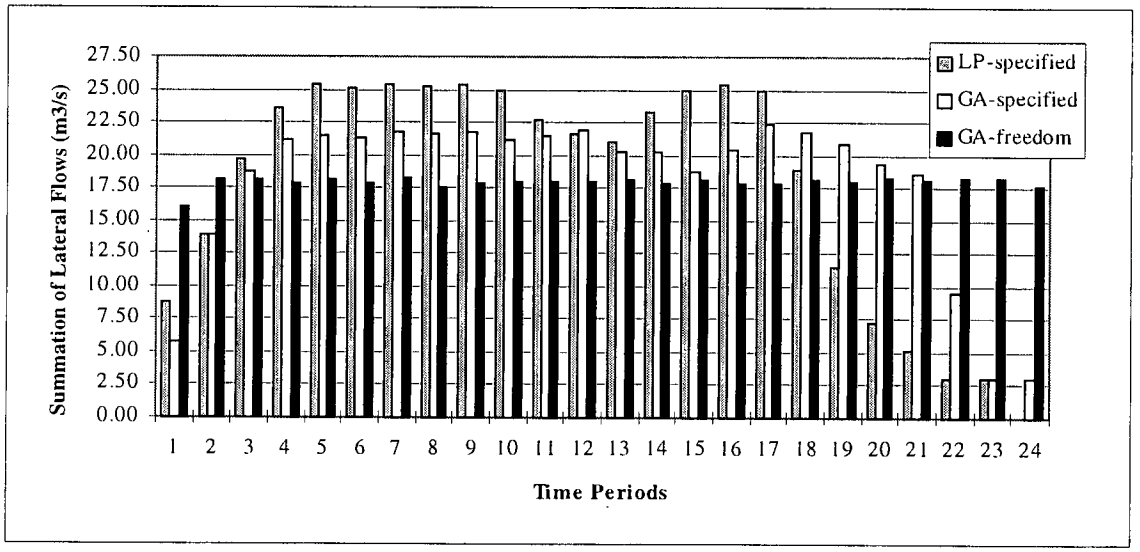

Figure 9.1 Comparison on summation of lateral flows in each time period

Additional model runs were carried out to determine if results were sensitive to the time step used. Runs were carried out with time steps of 6 hours and 24 hours. Results were very similar. The average outflow for the 6hour and 12 hour time steps was 4.7 m^3/s, while for the 24 hour time step, the average outflow was reduced to 4 m^3/s. With the 24 hour time step, water is used less effectively within the system. Starting time blocks determined were different, depending upon the time step used. Table 9.4 summarises results. Constraints were satisfied in all cases. The only significant impact of time step was in execution time. It took 16 seconds to run the GA with a 24 hour time step, 26 seconds with a 12 hour time step and 91 seconds with a 6 hour time step.

Both GA formulations were used with a population size of only 8, with a crossover probability of 0.85 and 0.15 mutations per chromosome. The GA generates a number of feasible solutions to the problem. The average fitness in the last generation was 534 and the fittest solution was 530. This indicates robustness in the approach and the possibility of adopting different alternatives.

Table 9.4 Starting time blocks with different time steps

Secondary Number	Starting time blocks		
	12-hr time block	6-hr time block	24-hr time block
1	4	8	2
2	6	12	3
3	4	8	2
4	4	8	2
5	6	12	3
6	3	6	1.5
7	7	14	3.5
8	3	6	1.5
:	:		
:	:		
114	5	10	2.5
115	5	10	2.5
116	1	2	0.5
117	5	10	2.5

9.6 Sensitivity Analyses

Sensitivity analyses have been carried out using the 12-hr time block with free selection of the starting time period. It has been found that the fittest chromosome produced the best result at crossover probability of 0.85, and 0.15 mutations per chromosome. However, the sensitivity to mutation and crossover probability was low. Figure 9.2 presents the sensitivity to crossover, and Figure 9.3 shows sensitivity to mutation. Good results can be produced over a wide range, and clearly solutions are robust.

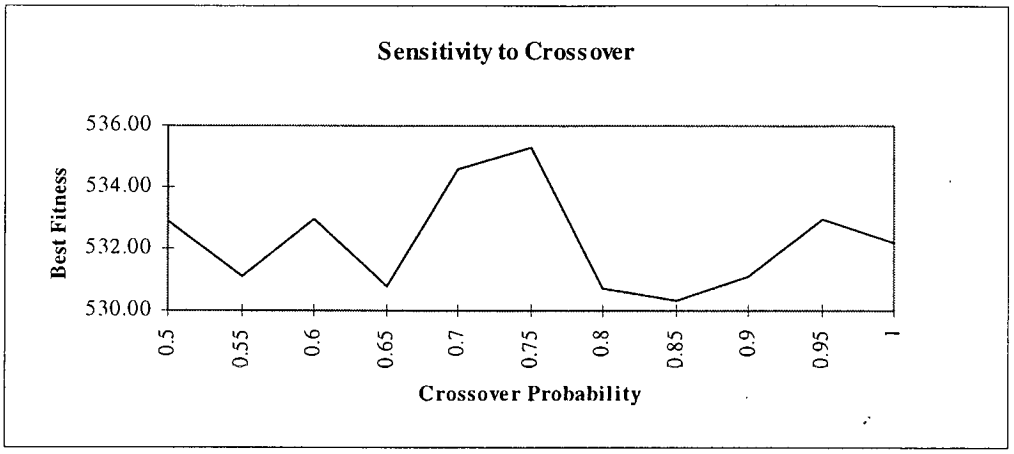

Figure 9.2 Sensitivity to crossover

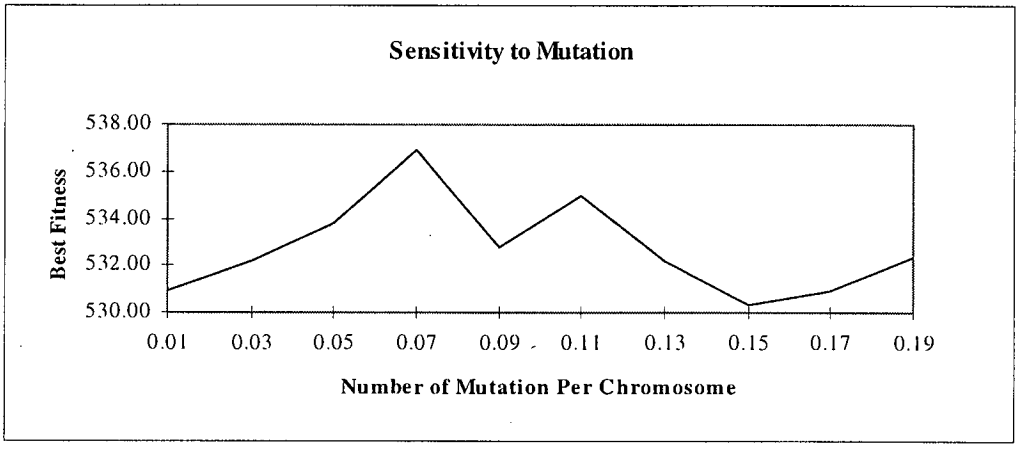

Figure 9.3 Sensitivity to mutation

Sensitivity to population size was also considered. Figure 9.4 present the sensitivity to population size. Increasing the population size beyond 8 did not result in any significant improvement in fitness. The reason for this is the simplicity of the objective function, and the penalty function used. Table 9.5 summarises the number of generations and execution times required for different population sizes.

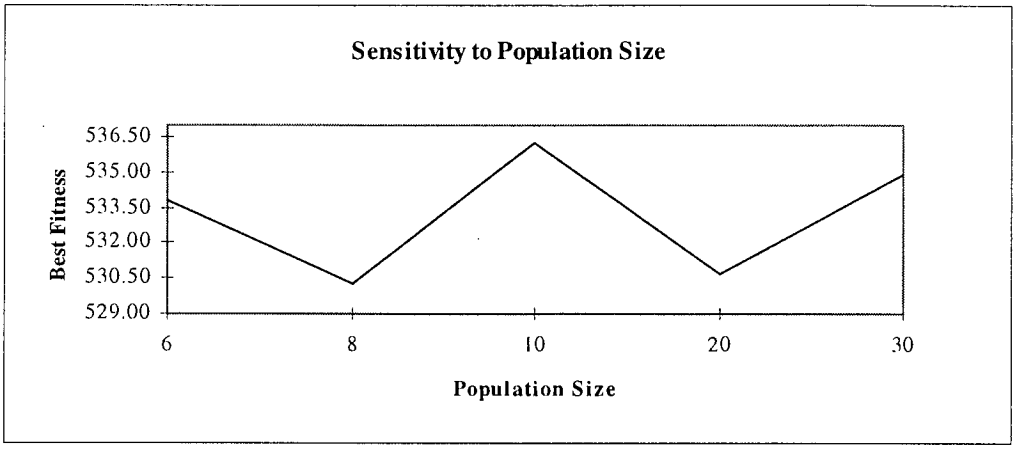

Figure 9.4 Sensitivity to population size

Table 9.5 Number of generations required for different population sizes

Population Size	**6**	**8**	**10**	**20**	**30**
Generation required	2400	1500	1200	1200	1500
Execution time (sec)	32.13	25.82	23.90	55.42	82.44

9.7 Conclusions

It is concluded that the GA is particularly well suited to this problem. It is robust and reaches the optimum very quickly. The formulation was easily set up, is very flexible and suitable to the problem. The GA performs significantly better than the 0-1 LP approach of Reddy et al. (1999), which was unable to satisfy the supply canal capacity constraint. The GA produces a number of candidate solutions and can be used to determine an optimal starting period for each lateral, or to determine the optimum starting period within a pre-specified range.

10. CONCLUSIONS AND RECOMMENDATIONS

10.1 Introduction

The focus of this research has been the investigation and development of GAs for real time irrigation water allocation, and irrigation water scheduling problems. This research presents some of the first applications of GAs in these areas, and there will clearly be scope for further development and improvements of the approach. In this chapter, the conclusions of the research are summarised and recommendations made for further investigation.

10.2 Principal Achievements

It has been demonstrated that in general a GA is relatively simple to set up and can be applied with non-linear and complex objective functions and constraints. Two types of optimisation problem have been investigated, and the main findings from research of these are summarised below.

10.2.1 Water Allocation Problem

- In the water allocation problem, the GA should be set up with canal flows as the decision variables. The number of decision variables can be reduced by making use of continuity at junctions with no irrigation demands to permit one canal flow to become a dependent variable.

- It is concluded that a real-value representation, proportional selection together with the elitist strategy, uniform crossover and non-uniform mutation will produce the best results. A crossover probability of 0.95 and 0.75 mutations per chromosome are suitable for the Tukad Ayung irrigation system in Bali. The approach has been shown to be robust and not particularly sensitive to crossover or mutation probabilities. It is sensitive to population size, however, and too small a population restricts diversity.

- In the water allocation problem, maintaining nodal water balances was found to be difficult. Performance of the GA was significantly improved by making one reach flow a dependant variable at nodes with no irrigation demands.

- It has been demonstrated that with the improved GA formulation, solutions to the water allocation problem can be achieved that are similar to those achieved by QP.

- The main advantage of the GA approach is that it can be set up with any form of objective function. However, for the water allocation problem, although the improved GA produced acceptable results, it offers no real advantage over the QP approach. Execution times for the QP approach were significantly shorter than those of the improved GA.

10.2.2 Water Scheduling Problems

- In canal water scheduling problems, a binary representation of canal opening and closure periods has been found to provide the most appropriate decision variables for the problem.

- A Zero-1 approach was the most appropriate of the formulations researched in this study, providing better performance than a warabandi approach that was also represented in a GA.

- It has been found that scheduling to a level below secondary canals can be achieved, but the water allocation is less equitable than could be produced by scheduling to secondary level only.

- A comparison with published results for a secondary canal scheduling problem solved using a linear programming approach (Reddy et al. 1999), demonstrated that the GA approach was more flexible, was apparently faster, produced better results in terms of an equitable distribution of resources.

- The GA approach is very well suited to scheduling problems, and as a part of this research a transparent set of objective functions and constraints for such problems have been developed.

10.3 Limitations of the GA Approach

The limitations of the GA approach may be summarised as follows:

- Setting the upper and lower bounds of the search space for decision variables is extremely important. If the these bounds are narrow and close to the acceptable solution the execution time is low. Conversely, if the bounds are wide a greater number of generations are required to find an acceptable solution and consequently execution times are much longer.

- In the water allocation problem it was found that binary representation was not capable of producing accurate results. To improve accuracy required longer string lengths, but as the length of a string increases, the chances of good solutions being disturbed increases. It was found that real value representation resulted in shorter strings and in improved solutions.

- The GA is clearly sensitive to string length, even in real value representation, and this will limit its application to larger problems. As string length increases, maintaining good solutions becomes more difficult.

- In the application of binary representation to the water allocation problem, genes have to be converted to the values of the decision variable by linear mapping, and this process slows down execution times. In the water scheduling problem binary representation was suitable for zero-1 approach, and required no mapping. The most appropriate representation scheme thus depends on the nature of the problem being investigated.

- In GAs technique, starting values used to search for optimum such as population size, crossover and mutation probabilities are in wide ranges. Table 10.1 summarises good starting values for trial in convergence to the best solution. However, these may not be the cases in other applications.

Table 10.1 Good starting values for trial to convergence.

Item	Trial values	Increment used
Population size	100 to 400*	100
Crossover probability	0.65 to 0.95	0.05
Number of mutation per chromosome	0.02 to 1.5	0.01

* In a very simple system, try 8 to 50.

10.4 Recommendations for Further Research

10.4.1 Hybrid GAs

In the application of a GA to the water allocation problem, it was found that the execution time was much higher than that required with a quadratic programming approach. It has been noticed that the GA converges rapidly towards a solution at first, but that refinement of solutions can take a large number of generations. Combining a GA with other local search method could produce more efficient search algorithm (Sharif 1999). A GA is good at identifying solutions close to the optimum because it searches from a population of points, not a single point. From the review of combined method presented in chapter 3, it would appear that combination of a GA with SA may offer some potential and would be worthy of further investigation in application to water resources problems.

10.4.2 Adaptation of GA Operators

GAs have been in use for more than 30 years. During this time there has been continuous development and adaptation of the operator techniques to improve rates of convergence. In the early applications of GAs, conventional operators were proportional selection, binary representation, conventional mutation and 1-site crossover. Various improved techniques have been developed such as tournament selection, real value representation, uniform and non-uniform crossovers. It is clear that there will be further possible ways that the GA approach could be improved through introducing new adaptations of operators. Further investigation of techniques that could preserve the best parts of solutions would be worthwhile.

REFERENCES

Ahuja, L.R., and Nielsen, D.R. (1990). "Field Soil-Water Relations." Irrigation of Agricultural Crops. *Agronomy. Monograph No.30. American Society of Agronomy*, WI. 143-190.

Allen, G. R., Pereira, L.S., Raes, D., and Smith, M. (1998). "Crop evapotranspiration", Guidelines for computing crop water requirements. FAO, Irrigation and Drainage paper 56.

Alway, F.J., and McDole, G.R. (1917). "Relation of water-retaining capacity of soil to its hygroscopic coefficient." *J. Agric. Res.*, 9, 27-71.

Atiya, A.F., ELShoura, S.M., Shaheen, S.I, and ELSherif, M.S. (1999). A comparison between neural-network forecasting techniques Case study:River flow forecasting. *IEEE Transactions on Neural Networks* Vol.10, No.2, pp.402-409.

Anwar A.A., and Clarke D. (2001). "Irrigation scheduling using mixed-integer linear programming". *J. Irri. and Drainage Engrg,* ASCE 127 (2) 63-69.

Bailey, R. (1990). *Irrigated crops and their management.* Farming Press Books, Ipswich, UK.

Beasley, D., and Heitkotter, J. (1995). "The Hitcher-Hikers guide to evolutionary computation",<NEWS_811586290@cm.cf.ac.uk>, Dept. of Comp. Sci., Univ. of Wales.

Bellman, R. (1957). *Dynamic Programming*, Princeton University Press, USA.

Bhuiyan, S.I. (1981). "Experience in field research in irrigation system management in the Philippines-some methodological issues". Paper presented at the International Seminar on Field Research Methodologies for Improved Irrigation System management, Coimbatore, Tamilnadu Agril. University, College of Agric. Engrng., Sept. 15-18.

Bottrall, A. (1981). "Action research towards improved water distribution". Paper presented at the International Seminar on Field Research Methodologies for Improved Irrigation Systems Management, Coimbatore, Tamil Nadu Agril. University, College of Agric. Engrg., Sept. 15-18.

Brisson, N. (1998). "An analytical solution for the estimation of the critical available soil water fraction for a single layer water balance model under growing crops". *Hydrol.and Earth Syst. Sci.* 2(2-3), 221-231.

Bruce, R.R., Dane, J.H., Quisenberry, V.L., Powell, N.L., and Thomas A.W. (1983). "Physical characteristics of soils in the southern region." *Cecil. Georgia Agric. Exp. Stn. South. Coop. Ser. Bull.* 267.

Burch, G.J., Smith, R.C.G. and Mason, W.K. (1978). "Agronomic and physiological responses of soybean and sorghum crops to water deficits. II crop evapotranspiration, soil water depletion and root distribution". *Aust. J. Plant Physiol.*, 5, 169-177.

Chen, Y.M. (1997). "Management of water resources using improved genetic algorithms". *J.Comp. and Electr. in Agric.* 18, 117-127.

Cieniawski, S.E., Eheart, J.W., and Ranjithan, S. (1995). "Using genetic algorithms to solve a multiobjective groundwater monitoring problem". *J. Water Resour. Res.* 31(2), 399-409.

Coals, A.V., and Wardlaw, R.B. (1994). "Review of computational procedures in dynamic programming with applications in water resources". *Tech. Reprt No.CENV0194:* Dept. of Civ. and Envir. Engrg. The Univ. of Edinburgh.

Curwen, D., and Massie, L. R. (1985). "WISP-the Wisconsin irrigation scheduling program". Advances in evapotranspiration: Proc., Nat. Conf. on Adv. in Evapotranspiration, *Am. Soc. of Agric. Engrs.* ASAE, St. Joseph, Mich., 351-356.

Darwin, C. (1959). *The origin of species.* University of Pennsylvania Press, Philadelphia.

Davis, L. (1991). *Handbook of genetic algorithms.* Van Nostrand Renhold, NY.

De Jong, K.A. (1975). "An analysis of the behaviour of a class of genetic adaptive systems." Doctoral dissertation, Univ. of Michigan, Ann Arbor, Mich.

Doorenbos, J., and Kassam, A.M. (1979). "Yield response to water." *FAO Irrigation and drainage paper 33*, Food and Agriculture Organisation, United Nations, Rome.

Ehlers, W., Hamblin, A.P., Tennant, D. and van der Ploeg, R.R. (1991). "Root system parameters determining water uptake of field crops". *Irrig. Sci.*, 12, 115-124.

Gardner, W.R. (1960). "Dynamic aspects of water availability to plants." *Soil Sci.* 89. 63-73.

Goldberg, D.E., and Kuo, C.H. (1987). "Genetic algorithms in pipeline optimization." *J. Comp. in Civ. Engrg.,* ASCE, 1(2), 128-141.

Goldberg, D.E. (1989). *Genetic Algorithms in search optimization & machine learning.* Addison-Wesley, Reading, Mass. USA.

Goldberg, D.E., and K.Deb. (1991). "A Comparative analysis of selection schemes used in genetic algorithms". *Foundations of genetic algorithms*, Morgan Kaufman, San Mateo, Calif., 69-93.

Hales, L.A. (1994). "A model to determine optimal water management procedures for multi-variable irrigation systems". Ph.D. Thesis . Institute of Irrigation Studies. Fac. of Engrg. and Applied Sci.. Univ. of Southampton.

Hannan, T.C., and Coals, V.A. (1995). "Real-time water allocation for irrigation". *J.The Institu.of Water And Envir.* 9(1), 19-26.

Hansen, V.E., Israelsen O.W., and Stringham G.E. (1980). *Irrigation Principles and Practices.* John Wiley and Sons, NY.

Heidari, M., Chow, V.T., Kokotovic, P.V., and Meredith, D.D. (1971). "Discrete differential dynamic programming approach to water resources systems optimization." Water Resour. Res., 7(2), 273-283.

Hillel, D. (1980). *Applications of soil physics.* Academic Press, NY.

Holland, J.H. (1971). "Proc. and processors for Schemata. In: Jacks, EL. (Ed.), Associative information processing." Elsevir, New York, 127-146.

Holland, J.H. (1975). *Adaptation in natural and artificial systems.* MIT Press, Cambridge, Mass.

Hollstien, R.B. (1971). "Artificial genetic adaptation in computer control systems." Diss. Abstr. Int. B, 32(3), 1510.

Huang, M.W. and Arora, J.S. (1997). "Optimal design with discrete variables: Some numerical experiments" *International J for Numerical. Methods in Engrg.* 40(1), 165-188.

Huck, M.G., and Hillel, D. (1983). "A model of root growth and water uptake accounting for photosynthesis, respiration, transpiration, and soil hydraulics." *Adv. Irrig.* 2, 273-335.

Israelsen, O.W., and Hansen,V.E. (1962). *Irrigation Principles and Practices*, John Wiley and Sons Inc., USA.

Israelsen, O.W., and West, F.L. (1922). "Water holding capacity of irrigated soils." *Utah State Univ. Agric. Exp. Stn. Bull.* 183.

Jansson, A. (1994). "Fluid Power System Design -A Simulation Approach". Ph.D. Thesis. Div. of Fluid Power Technology, Dept. of Mech. Engrg, Linkoping Univ.

Jensen M.E., Rangeley W.R. and Dieleman P.J.(1990). "Irrigation Trends in World Agriculture". Irrigation of Agricultural Crops. *Agronomy. Monograph No.30. American Society of Agronomy*, WI., 31-67.

LINDO Systems Inc. (1995). *Super LINGO optimisation modelling package*. LINDO Systems Inc., Chicago.

Malhotra, S.P., Raheja S.K., and Seckler D. (1984). "A methodology for monitoring the performance of large-scale irrigation systems: A case study of the Warabandi system of northwest India." *J.Agri.Admin.* 17, 231-259.

Markowitz, H. (1959). *Portfolio Selection, Efficient Diversification of Investments*. New York: Wiley.

Michalewicz, Z.(1992). *Genetic algorithms + data structures = evolution programs*. Springer-Verlag, New York, Inc., New York.

Mott MacDonald (1998). *Supporting Report 3*, Canal Operation Studies: IGNP Studies for the State of Rajasthan. Draft Final Report. Cambridge.

Mott MacDonald (1999). Main Report:IGNP Studies for the State of Rajasthan. *Final Report*. Cambridge.

Murray, D., and Yakowitz, S.J. (1979). "Constrained differential dynamic programming and its application to multireservoir control, *Water Resour. Res.*, 15(4), 1017-1027.

Naadimuthu, G., Raju, K.S., and Lee, E.S. (1999). "A heuristic dynamic optimization algorithm for irrigation scheduling." *J. Math. and Comp. Modelling.*, 30(7-8), 169-183.

NAG (1995). *NAG C Library, Mark 3*, CLD0203DM, Microsoft Windows Dynamic Link Library(16-bit). Numerical Algorithms Group Ltd., Oxford.

Nofziger, D.L., Williams, J.R., Hornsby A.G., and Wood, A.L. (1983). "Physical characterisTics of soils of the southern region: Bethany, Konawa, and Tipton series." *Oklahoma State Univ. Agric. Exp. Stn. South. Coop. Ser. bull.* 265.

Ogata, G., and Richards, L.A. (1957). "Water content change following irrigation of bare field soil that is protected from evaporation." *Soil Sci.Soc.Am.* Proc. 21, 355-356.

Oliveira, R., and Loucks, D.P. (1997). "Operating rules for multireservoir systems." *Water Resour. Res.*, 33(4), 839-852.

Rao, N.H., Sarma, P.B.S., and Chander, S. (1992). "Real-time adaptive irrigation scheduling under a limited water-supply". *J. Agric. Water Mgmt.*, 20(4), 267-279.

Ratliff, L.F., Ritchie, J.T., and Cassel D.K. (1983). "Field-measured limits of soil water availability as related to laboratory measured properties". *Soil Sci. Soc. Am. J.* 47, 770-775.

Rawlins, S.L., and Raats, P.A.C. (1975). "Prospects for high frequency irrigation". Science 188, 604-610.

Reddy, J.M, Wilamowski, B., and Cassel-Sharmasarkar, F. (1999). "Optimal scheduling of irrigation for lateral canals". *ICID J.* 48(3).

Richards, L.A., Gardner, W.R., and Ogata, G. (1956). "Physical processes determining water loss from soils. *Soil Sci.Soc.Am.* Proc. 20, 310-314.

Rijov, S.N., Hagan R.M., and Aston M.J. (1973). "Water, Plant Growth and Crop Irrigation; Part D: Water requirement of crops and their determination" .*Irrigation/Drainage and Salinity.* An international source book, FAO. Hutchinson/FAO/Unesco. London.

Savic, D.A., and Walters, G.A. (1997). "Genetic algorithms for least-cost design of water distribution networks." *J. Water Resour. Plang. and Mgmt.*, ASCE, 123(2), 67-77.

Schrage, L. (1986). *User's Manual for LINDO.* Palo Alto, Calif.: Scientific Press.

Sharif, M. (1999). "Multi reservoir systems optimisation using genetic algorithms." PhD Thesis, School of Civil and Environmental Engineering, The Univ. of Edinburgh, Edinburgh.

Shrestha, D.K. (1999). "A model study to optimise the warabandi system by introducing crop based irrigation." MSc dissertation, Institute of Irrigation and Development Studies, Univ. of Southampton, Highfield, Southampton.

Shyam. R., Chauhan H.S., and Sharma. J.S. (1994). "Optimal operation scheduling model for a canal system". *J. of Agric. Water Mgmt.*, 26, 213-225.

Singh. B., Boivin, J., Kirkpatrick, G., and Hum, B. (1995). "Automatic Irrigation Scheduling System(AISSUM): Principles and Applications". *J. Irri. and Drainage Engrg.* ASCE 121(1), 43-56.

Sir M MacDonald and Partners Asia. (1989). "Bali Water Resources Study; resources development planning for Denpasar" *DGWRD, Jakarta.*

Sir M MacDonald and Partners (1988). " East Java provincial water resources master plan study for portable water supply". *DGWRD, Jakarta.*

Spiertz, H.L.J. (1991). "The transformation of traditional law: a tale of people's participation in irrigation management on Bali". *Landscape and Urban Plang.,* Elsevier Science Publishers B.V., Amsterdam. 20, 189-196.

Sunantara, J.D., and Ramirez, J.A. (1997). "Optimal stochastic multicrop seasonal and intraseasonal irrigation control." *J. Water Resour. Plang. and Mgmt.,* 123(1), 39-48.

Suryavanshi A.R., and Reddy J.M. (1986). "Optimal operation schedule of irrigation distribution systems". *J. Agric. Water Mgmt.*, 11, 23-30.

Svehlik, Z.J. (1987). "Estimation of irrigation system capacity". In: J.R. Rydzewski (Editor), Irrigation Development Planning: An Introduction for Engineers. John Willey and Sons, Chichestor.

Turgeon, A. (1982). "Incremental dynamic programming may yield non-optimal solutions." Water Resour. Res., 18(6), 1599-1604.

Vanclooster, M., Viaene, P., Christiaens, K. and Duchegre, S. (1996). "WAVE, Water & Agrochemicals in Soil and Vadose Environment". Institue for Land & Water management, Catholoic University of Leuven, Leuven, Belgium.

Veihmeyer, F.J., and Hendrickson, A.H. (1927). "Soil moisture conditions in relation to plant growth." *Plant Physiol.* 2, 71-78.

Veihmeyer, F.J., and Hendrickson, A.H. (1931). "The moisture equivalent as a measure of the field capacity of soils." *Soil Sci.* 32, 181-193.

Veihmeyer, F.J., and Hendrickson, A.H. (1949). "Methods of measuring field capacity and wilting percentages of soils." *Soil Sci.* 68, 75-94.

Veihmeyer, F.J., and Hendrickson, A.H. (1950). "Soil moisture in relation to plant growth." *Annu. Rev. Plant Physiol.* 1, 285-304.

Veihmeyer, F.J., and Hendrickson, A.H. (1955). "Does transpiration decrease as the soil moisture decreases?" *Trans. Am. Geophys. Union* 36, 425-448.

Wang, J., Hou, T., Chen, L, and Xu, X. (1999). "Conformational analysis of peptides using Monte Carlo simulations combined with the genetic algorithm." *J. Chemometrics and Intelligent Lab. Systems.* 45(1-2), 347-351.

Wang, Q.J. (1991). "The genetic algorithm and its application to calibrating conceptual rainfall-runoff models." Water Resour. Res., 27(9), 2467-2471.

Wang, M., and Zheng, C. (1998). "Ground water management optimization using genetic algorithms and simulated annealing : Formulation and comparison". *J. of the Am. Water Resour. Assoc.* 34(3), 519-530.

Wang, Z., Reddy, J.M., and Feyen, J. (1995). "Improved 0-1 programming model for optimal flow scheduling in irrigation canals". *J. Irrigation and Drainage Systems* Kluwer Academic Publishers. 9, 105-116.

Wardlaw, R.B., and Coals, A.V. (1994). "Assessment of appropriate algorithms for optimal allocation of irrigation water in real time". *Report Number CENV0294*, Dept. of Civ. and Envir. Engrg, The Univ. of Edinburgh.

Wardlaw, R.B., and Barnes, J.M. (1996). "Improved irrigation system planning and management, Optimal allocation of irrigation water supplies." *ODA TDR Research Project No.6261, Phase I Rep.*

Wardlaw, R.B., and Barnes, J.M. (1997). "Improved irrigation system planning and management, Optimal allocation of irrigation water supplies." *ODA TDR Research Project No.6261, Phase II Rep.*

Wardlaw, R.B., and Barnes, J.M. (1998). "Improved irrigation system planning and management, Optimal allocation of irrigation water supplies." *DFID TDR Research Project No.6261, Phase III Rep.*

Wardlaw, R.B., and Barnes, J.M. (1999). "Optimal allocation of irrigation water supplies in real time." *J. Irri. and Drainage Engrg.*, ASCE 125(6), 345-354.

Wardlaw, R.B., Moore, D.N., and Barnes, J.M. (1997). "An assessment of the potential of optimisation in real time irrigation management". *Dept. of Civil and Environmental Engineering, the University of Edinburgh.*

Wardlaw, R.B., and Sharif, M. (1999). "Evaluation of genetic algorithms for optimal reservoir system operation". *J. Water Resour. Plng. and Mgmt.*, ASCE 125(1), 25-33.

Wardlaw, R.B., and Wells, R.J. (1996). "Simulating crop production and resource development impacts: the Lower Ayung Simulation Model." *Proc. Instn. Civ. Engrs Water, Marit.&Energy.* 118(2).

Warrick, A.W. (1990). "Nature and Dynamics of Soil Water". Irrigation of Agricultural Crops. *Agronomy. Monograph No.30. American Society of Agronomy.* 69-92.

Whitley, D. (1989). "The GENITOR algorithm and selection pressure: why rank best allocation of reproductive trials is best," Proc. 3rd Int Conf. On Genetic Algorithms, J.D. Schaffer, ed., Morgan Kaufman, San Francisco.

Winston, W.L. (1994). *Operations research: application and algorithms.* Third Edition. Indiana Univ.

Yang, C.C., Prasher, S.O., and Lacroix, R. (1996). "Artificial Nueral Networks to land drainage engineering". *Transaction of the ASAE* Vol.39, No.2, pp 525-533.

Yang, C.C., Prasher, S.O., Lacroix, R., Sreekanth, S., Patni, N.K., and Masse, L. (1997). "Artificial neural network model for subsurface-drained farmlands". *J.Irrig. and Drainge Engrg.* ASCE 123(4), 285-292.

Yang, J. P., and Soh, C.K. (1997). "Structural optimization by genetic algorithms with tournament selection." *J. of Comp.in Civ..Engrg.,* ASCE, 11(3), 195-200.

APPENDIX A DATA FILES FOR WATER ALLOCATION TEST SYSTEM

Table A-1 shows the descriptions of the network data file which was used to model the simple canal system. The network file describes the irrigation system. The first part is comprised the reach information, outlines of connection and the channel capacity. The second part describes the node type that could either be an inflow node, irrigation node or sink (outflow) node.

The inflow file and the irrigation demand file are set up in two dimensional arrays. Reach number, reach capacity and connection index was set differently from node to node (see Table A-2 and A-3). The demand file shows the demand in node 1,2,3,4,5,6 as 0,4,5,5,0 and 3 units respectively. Inflow to node no.1 is 15 units.

A network file for a more complex canal system of water allocation problem is shown in Table A-4. Demand file is in Table A-5 and Inflow file is in Table A-6.

Table A-1 Test network array data file for a simple network

Data file	**Comments**
Test Network	title line
6	number of nodes
1 2 3 4 5 6	node number
1	number of sink
5	sink number
4	number of irrigation schemes
2 3 4 6	irrigation scheme number
Connection index	
1 2 3 4 5 6	node number
1 0 -1 -1 0 0 0	flow direction
2 1 0 0 0 0 -1	negative indicate from i to j
3 1 0 0 -1 0 0	where,

4	0	0	1	0	-1	0	i is the row
5	0	0	0	1	0	1	j is the column
6	0	1	0	0	-1	0	
Reach Capacities							
	1	2	3	4	5	6	node number
1	0	15	15	0	0	0	reach capacity in cms
2	15	0	0	0	0	15	
3	15	0	0	15	0	0	
4	0	0	15	0	15	0	
5	0	0	0	15	0	15	
6	0	15	0	0	15	0	
Reach Numbers							
	1	2	3	4	5	6	node number
1	0	1	2	0	0	0	reach number which link between nodes
2	1	0	0	0	0	3	
3	2	0	0	4	0	0	
4	0	0	4	0	5	0	
5	0	0	0	5	0	6	
6	0	3	0	0	6	0	

Table A-2 Test demand file for a simple network

Data File	**Comments**
Test Demand File	
1	number of times step
6	number of nodes
1 2 3 4 5 6	node number
0 4 5 5 0 3	node demand

Table A-3 Test inflow file for a simple network

Data File	**Comments**
Test Inflow File	Title
1	number of years
6	number of nodes
1 2 3 4 5 6	node number
15 0 0 0 0 0	inflow (units)

Table A-4 Test network array data file for a more complex network

Data file	**Comments**
Test Network	title line
10	number of nodes
1 2 3 4 5 6 7 8 9 10	node number
1	number of sink
10	sink number
8	number of irrigation schemes
1 2 3 5 6 7 8 9	irrigation scheme number
Connection index	
1 2 3 4 5 6 7 8 9 10	node number
1 0 -1 -1 0 0 0 0 0 0 0	flow direction
2 1 0 0 -1 0 0 0 0 0 0	negative indicate from i to j
3 1 0 0 -1 -1 0 0 0 0 0	where,
4 0 1 1 0 0 -1 -1 0 0 0	i is the row
5 0 0 1 0 0 0 -1 0 -1 0	j is the column
6 0 0 0 1 0 0 0 0 0 0	
7 0 0 0 1 1 0 0 -1 0 0	
8 0 0 0 0 0 0 1 0 1 -1	
9 0 0 0 0 1 0 0 -1 0 0	
10 0 0 0 0 0 0 0 1 0 0	

Reach Capacities

	1	2	3	4	5	6	7	8	9	10	
											node number
1	0	3	15	0	0	0	0	0	0	0	reach capacity in cms
2	3	0	0	8	0	0	0	0	0	0	
3	15	0	0	3	8	0	0	0	0	0	
4	0	8	3	0	0	2.5	5	0	0	0	
5	0	0	8	0	0	0	2	0	3	0	
6	0	0	0	2.5	0	0	0	0	0	0	
7	0	0	0	5	2	0	0	4	0	0	
8	0	0	0	0	0	0	4	0	2	2	
9	0	0	0	0	3	0	0	2	0	0	
10	0	0	0	0	0	0	0	2	0	0	

Reach Numbers

	1	2	3	4	5	6	7	8	9	10	
											node number
1	0	1	2	0	0	0	0	0	0	0	reach number which link between nodes
2	1	0	0	3	0	0	0	0	0	0	
3	2	0	0	4	5	0	0	0	0	0	
4	0	3	4	0	0	6	7	0	0	0	
5	0	0	5	0	0	0	8	0	9	0	
6	0	0	0	6	0	0	0	0	0	0	
7	0	0	0	7	8	0	0	10	0	0	
8	0	0	0	0	0	0	10	0	11	12	
9	0	0	0	0	9	0	0	11	0	0	
10	0	0	0	0	0	0	0	12	0	0	

Table A-5 Test demand file for a more complex test system

Data File	**Comments**
Test Demand File	
1	number of time step
10	number of nodes
1 2 3 4 5 6 7 8 9 10	node number
1.1 0.9 5.8 0 3.5 2.5 1.2 1 1 0	node demand

Table A-6 Test inflows file for a more complex test system

Data File	**Comments**
Test Inflow File	Title
1	number of years
10	number of nodes
1 2 3 4 5 6 7 8 9	node number
14 2 0 0 0 0 0 0 0	inflow (units)

APPENDIX B AYUNG IRRIGATION SYSTEM DATA FILES

This appendix contains network, file demand file and inflows file used for Ayung irrigation system. In Ayung network file part of "Calculated Reach Index" is extension part from Ayung network file of Wardlaw and Barnes (1996). The reach calculated index is the index for the computed reach for any juction node.

Demand file of Ayung Irrigation system is shown below after Ayung network file respectively.

AYUNG Network File (Wardlaw and Barnes, 1996)

Title
56
3 4 6 106 7 10 11 14 15 18 21 22 122 23 24 27 127 28 32 35 36 40 41 43 44 45 46 50 51 54 55 57 59 61 62 64 65 66 69 70 71 72 74 76 77 78 79 81 82 83 86 87 88 89 92 99
10
24 36 66 72 78 79 83 89 92 99
13
6 106 22 122 27 127 45 50 65 71 77 81 88

Connection Index
3 4 6 106 7 10 11 14 15 18 21 22 122 23 24 27 127 28 32 35 36 40 41 43 44 45 46 50 51 54 55 57 59 61 62 64 65 66 69 70 71 72 74 76 77 78 79 81 82 83 86 87 88 89 92 99
3 0 -1 0 0 0

36 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
40 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 -1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
41 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 -1 0 0 0
0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
43 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
44 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 -1 0 0 0 -1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
45 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 -1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
46 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
-1 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
50 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 -1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
51 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 -1 -1
0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
54 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0
-1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
55 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
-1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
57 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1
0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
59 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0
61 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0
0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
62 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1
64 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 -1 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
65 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
66 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
69 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 1 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
70 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 -1 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
71 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
72 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
74 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 -1 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
76 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 -1 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
77 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
78 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
79 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
81 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

82 0 0 0

0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
18 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 7.00 0.00 100.00 0.00 0.00 0.00 0.00
0.00 100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
21 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 100.00 0.00 0.00 100.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
22 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 100.00 100.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
122 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 100.00 100.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
23 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 100.00 0.00 0.00 -0.65
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 -0.02 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 -0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
24 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 -0.65 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
27 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 100.00 100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
127 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 100.00 0.00 0.00 0.00 0.00 0.00
100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
28 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
100.00 0.00 0.00 -0.10 0.00 0.00 0.00 0.00 -0.29 0.00 0.00 0.00 0.00 0.00 -0.61 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
32 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 8.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 -0.10 0.00 0.00 0.00 100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
35 0.00 0.00 0.00 0.00 -0.97 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 -0.93 -0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 -0.06 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
36 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 -0.93 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
40 0.00 0.00 0.00 0.00 0.00 0.00 100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 100.00 -0.01 0.00 0.00 100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
41 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 100.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00

0.00 0.00 100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
43 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 -0.02 0.00
0.00 0.00 -0.29 0.00 0.00 0.00 0.00 0.00 0.00 100.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
44 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 100.00 0.00 1.00 0.00 0.00 0.00 100.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
45 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 1.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
46 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 -
0.02 -0.98 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
50 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
51 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 -0.19 -0.54
0.00 0.00 -0.27 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
54 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 -0.61 0.00 0.00 0.00 0.00 0.00 0.00 100.00 0.00 0.00 0.00 -0.19 0.00 0.00
100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
55 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 -0.54 0.00 0.00
100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
57 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 -0.02 0.00 0.00 100.00 100.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
59 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 -0.33 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 -0.98 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 100.00 0.00 0.00 0.00 0.00 0.00
61 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 -0.06 0.00 0.00 100.00 0.00 0.00 0.00 0.00 0.00 -0.27 0.00 0.00
0.00 0.00 0.00 100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
62 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 100.00 0.00 100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 100.00
64 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 100.00 0.00 4.00 0.00 5.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
65 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

0.00 0.00 0.00 0.00 4.00 0.00 4.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
66 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 4.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
69 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
100.00 0.00 0.00 0.00 5.00 0.00 0.00 0.00 100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
70 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 100.00 0.00 100.00 0.00 100.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
71 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 100.00 0.00 100.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
72 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 100.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
74 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 100.00 0.00 0.00 0.00 100.00 0.00 0.00 0.00
100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
76 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 100.00 0.00 100.00 0.00 100.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
77 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 100.00 0.00 100.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
78 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 100.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
79 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 100.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
81 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 100.00 0.00 0.00 0.00 0.00
0.00 100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
82 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
100.00 0.00 -0.73 -0.27 0.00 0.00 0.00 0.00 0.00
83 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 -0.73 0.00 0.00 0.00 0.00 0.00 0.00 0.00
86 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

0.00 100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 -0.27 0.00 0.00 100.00 0.00 0.00 0.00 0.00
87 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 100.00 0.00 10.00 0.00 10.00 0.00
88 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 10.00 0.00 100.00 0.00 0.00
89 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 100.00 0.00 0.00 0.00
92 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 10.00 0.00 0.00 0.00 0.00
99 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

Reach Numbers

3 4 6 106 7 10 11 14 15 18 21 22 122 23 24 27 127 28 32 35 36 40 41 43 44 45
46 50 51 54 55 57 59 61 62 64 65 66 69 70 71 72 74 76 77 78 79 81 82 83 86 87 88
89 92 99

3 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

4 3 0 0 5 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

6 0 0 0 106 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

106 0 5 106 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

7 0 0 6 0 0 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

10 0 4 0 0 8 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

11 0 0 0 0 0 0 10 0 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 11 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

14 0 0 0 0 0 0 0 12 0 14 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

15 0 0 0 0 0 0 0 0 0 14 0 15 0 0 0 0 0 0 0 0 0 16 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

18 0 0 0 0 0 0 0 0 0 15 0 19 0 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

21 0 0 0 0 0 0 0 0 0 0 0 19 0 0 21 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

22 0 0 0 0 0 0 0 0 0 0 0 0 0 0 122 22 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

122 0 0 0 0 0 0 0 0 0 0 0 21 122 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

23 0 0 0 0 0 0 0 0 0 0 0 0 22 0 0 24 0 0 0 0 0 0 0 0 23 0 0 0 0 0 0 0
0 0 25 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

24 0 0 0 0 0 0 0 0 0 0 0 0 0 0 24 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 127 27 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
127 0 0 0 0 0 0 0 0 0 18 0 0 0 0 0 0 127 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
28 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 27 0 0 29 0 0 0 0 0 28 0 0 0 0 0 0 30
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
32 0 0 0 0 0 0 0 0 0 16 0 0 0 0 0 0 0 0 0 29 0 0 0 0 32 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
35 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 35 37 0 0 0 0 0 0 0 0 0 0
0 0 34 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
36 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 35 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
40 0 0 0 0 0 0 0 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 32 37 0 0 40 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
41 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 40 0 0 0 0 0 0 42 0 0 0 0
0 0 41 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
43 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0 0 0 28 0 0 0 0 0 0 0 43 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
44 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 43 0 0 44 0 0 0 0 48
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
45 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 44 0 0 45 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
46 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 45 0 0 0 0 0 0
46 47 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
50 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 42 0 0 0 0 0 0 50 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
51 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 50 0 52
53 0 0 51 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
54 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 30 0 0 0 0 0 0 0 48 0 0 0 0 52 0
0 54 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
55 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 53 0 0
55 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
57 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 46 0 0 54
55 0 0 0 0 0 0 0 0 57 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
59 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 0 0 0 0 0 0 0 0 0 0 0 0 0 47 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
61 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 34 0 0 41 0 0 0 0 0 51 0
0 0 0 0 0 61 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
62 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 61 0 63 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 62
64 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 63 0 64 0 66 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
65 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 64 0 65 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
66 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 65 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
69 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
57 0 0 0 66 0 0 0 69 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
70 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 69 0 70 0 72 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
71 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 70 0 71 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
72 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 71 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
74 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 72 0 0 0 74 0 0 0 75 0 0 0 0 0 0 0 0 0 0

76 0 0 0

23 0 0 0

72 0 0 0

Yield response factors (k)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongon										
1	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33	0.00	1.87
	1.87	3.25	1.80	1.80	1.00	0.00	0.75	0.75	1.20	1.20
	1.20	2.50	2.50	0.33						
2	2.50	0.33	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33
	0.00	1.87	1.87	3.25	1.80	1.80	1.00	0.00	0.75	0.75
	1.20	1.20	1.20	2.50						
3	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50
	2.50	0.33	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50
	0.33	0.00	0.00	0.00						
4	0.33	0.00	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20
	1.20	2.50	2.50	0.33	0.00	0.00	0.00	0.75	0.75	1.20
	1.20	1.20	2.50	2.50						

SCHEME 6A : CROP PATTERN 1: KADEWATAN

106

4

Gross crop water requirements (m^3/s)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongon										
1	1.084	1.084	0.566	0.856	0.380	0.671	0.341	0.000	0.000	0.050
	0.424	0.609	0.631	0.508	0.254	0.000	1.178	1.178	0.751	1.042
	0.784	1.074	0.767	0.000						
2	0.657	0.000	0.973	0.973	0.402	0.693	0.383	0.674	0.496	0.000
	0.000	0.238	0.460	0.631	0.598	0.486	0.301	0.000	1.178	1.178
	0.809	1.099	0.821	1.111						
3	0.000	0.000	0.000	0.000	0.829	0.829	0.404	0.695	0.541	0.832
	0.605	0.000	0.000	1.146	1.162	0.658	0.994	0.683	1.020	0.684
	0.000	0.000	0.000	0.000						
4	0.000	0.000	0.000	0.000	0.000	0.000	0.859	0.859	0.564	0.854
	0.650	0.940	0.620	0.000	0.000	0.000	0.000	1.178	1.178	0.751
	1.099	0.784	1.111	0.767						

Yield response factors (k)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongon										
1	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33	0.00	1.87
	1.87	3.25	1.80	1.80	1.00	0.00	0.75	0.75	1.20	1.20
	1.20	2.50	2.50	0.33						
2	2.50	0.33	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33
	0.00	1.87	1.87	3.25	1.80	1.80	1.00	0.00	0.75	0.75
	1.20	1.20	1.20	2.50						
3	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50
	2.50	0.33	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50
	0.33	0.00	0.00	0.00						
4	0.33	0.00	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20
	1.20	2.50	2.50	0.33	0.00	0.00	0.00	0.75	0.75	1.20
	1.20	1.20	2.50	2.50						

SCHEME 22B :CROP PATTERN 2 : MAMBAL

22
6
Gross crop water requirements (m^3/s)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongons										
1	0.902	0.902	0.470	0.712	0.316	0.558	0.284	0.000	0.000	0.041
	0.353	0.507	0.525	0.422	0.211	0.000	0.980	0.980	0.429	0.670
	0.456	0.698	0.442	0.000						
2	0.000	0.902	0.809	0.470	0.576	0.316	0.560	0.284	0.000	0.000
	0.198	0.353	0.525	0.525	0.404	0.211	0.000	0.000	0.000	0.000
	0.000	0.242	0.000	0.242						
3	0.000	0.000	0.809	0.809	0.335	0.576	0.319	0.560	0.412	0.000
	0.000	0.914	0.953	0.371	0.593	0.336	0.614	0.339	0.000	0.000
	0.000	0.000	0.000	0.000						
4	0.000	0.000	0.000	0.809	0.690	0.335	0.578	0.319	0.692	0.412
	0.000	0.000	0.000	0.953	0.966	0.351	0.631	0.372	0.652	0.373
	0.000	0.000	0.000	0.000						
5	0.000	0.000	0.000	0.000	0.690	0.690	0.336	0.578	0.450	0.692
	0.503	0.000	0.000	0.000	0.000	0.966	0.980	0.389	0.670	0.410
	0.698	0.415	0.000	0.000						
6	0.000	0.000	0.000	0.000	0.000	0.690	0.715	0.336	0.711	0.450
	0.782	0.503	0.000	0.000	0.000	0.000	0.000	0.980	0.980	0.429
	0.718	0.456	0.728	0.442						

Yield response factors (k)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongon										
1	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33	0.00	1.87
	1.87	3.25	1.80	1.80	1.00	0.00	0.75	0.75	1.20	1.20
	1.20	2.50	2.50	0.33						
2	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33	0.00
	1.87	1.87	3.25	1.80	1.80	1.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00						
3	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33
	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33	0.00
	0.00	0.00	0.00	0.00						
4	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50
	0.33	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50
	0.33	0.00	0.00	0.00						
5	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50
	2.50	0.33	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20
	2.50	2.50	0.33	0.00						
6	0.33	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20
	2.50	2.50	0.33	0.00	0.00	0.00	0.00	0.75	0.75	1.20
	1.20	1.20	2.50	2.50						

SCHEME 22A :CROP PATTERN 2 : MAMBAL
122
6
Gross crop water requirements (m^3/s)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongons										

1	0.738	0.738	0.385	0.582	0.259	0.456	0.232	0.000	0.000	0.034
	0.288	0.414	0.429	0.346	0.173	0.000	0.801	0.801	0.351	0.548
	0.373	0.571	0.362	0.000						
2	0.000	0.738	0.662	0.385	0.471	0.259	0.458	0.232	0.000	0.000
	0.162	0.288	0.429	0.429	0.330	0.173	0.000	0.000	0.000	0.000
	0.000	0.198	0.000	0.198						
3	0.000	0.000	0.662	0.662	0.274	0.471	0.261	0.458	0.337	0.000
	0.000	0.747	0.780	0.303	0.485	0.275	0.502	0.277	0.000	0.000
	0.000	0.000	0.000	0.000						
4	0.000	0.000	0.000	0.662	0.564	0.274	0.472	0.261	0.566	0.337
	0.000	0.000	0.000	0.780	0.790	0.287	0.516	0.304	0.533	0.305
	0.000	0.000	0.000	0.000						
5	0.000	0.000	0.000	0.000	0.564	0.564	0.275	0.472	0.368	0.566
	0.412	0.000	0.000	0.000	0.000	0.790	0.801	0.318	0.548	0.336
	0.571	0.339	0.000	0.000						
6	0.000	0.000	0.000	0.000	0.000	0.564	0.585	0.275	0.581	0.368
	0.640	0.412	0.000	0.000	0.000	0.000	0.000	0.801	0.801	0.351
	0.588	0.373	0.596	0.362						

Yield response factors (k)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongon										
1	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33	0.00	1.87
	1.87	3.25	1.80	1.80	1.00	0.00	0.75	0.75	1.20	1.20
	1.20	2.50	2.50	0.33						
2	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33	0.00
	1.87	1.87	3.25	1.80	1.80	1.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00						
3	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33
	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33	0.00
	0.00	0.00	0.00	0.00						
4	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50
	0.33	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50
	0.33	0.00	0.00	0.00						
5	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50
	2.50	0.33	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20
	2.50	2.50	0.33	0.00						
6	0.33	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20
	2.50	2.50	0.33	0.00	0.00	0.00	0.00	0.75	0.75	1.20
	1.20	1.20	2.50	2.50						

SCHEME 27B :CROP PATTERN 2 : MAMBAL

27

6

Gross crop water requirements (m^3/s)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongons										
1	0.281	0.281	0.147	0.222	0.099	0.174	0.089	0.000	0.000	0.013
	0.110	0.158	0.164	0.132	0.066	0.000	0.306	0.306	0.134	0.209
	0.142	0.218	0.138	0.000						
2	0.000	0.281	0.253	0.147	0.180	0.099	0.175	0.089	0.000	0.000
	0.062	0.110	0.164	0.164	0.126	0.066	0.000	0.000	0.000	0.000
	0.000	0.075	0.000	0.075						

3	0.000	0.000	0.253	0.253	0.104	0.180	0.099	0.175	0.129	0.000
	0.000	0.285	0.298	0.116	0.185	0.105	0.192	0.106	0.000	0.000
	0.000	0.000	0.000	0.000						
4	0.000	0.000	0.000	0.253	0.215	0.104	0.180	0.099	0.216	0.129
	0.000	0.000	0.000	0.298	0.301	0.110	0.197	0.116	0.203	0.117
	0.000	0.000	0.000	0.000						
5	0.000	0.000	0.000	0.000	0.215	0.215	0.105	0.180	0.140	0.216
	0.157	0.000	0.000	0.000	0.000	0.301	0.306	0.121	0.209	0.128
	0.218	0.129	0.000	0.000						
6	0.000	0.000	0.000	0.000	0.000	0.215	0.223	0.105	0.222	0.140
	0.244	0.157	0.000	0.000	0.000	0.000	0.000	0.306	0.306	0.134
	0.224	0.142	0.227	0.138						

Yield response factors (k)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongon										
1	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33	0.00	1.87
	1.87	3.25	1.80	1.80	1.00	0.00	0.75	0.75	1.20	1.20
	1.20	2.50	2.50	0.33						
2	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33	0.00
	1.87	1.87	3.25	1.80	1.80	1.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00						
3	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33
	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33	0.00
	0.00	0.00	0.00	0.00						
4	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50
	0.33	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50
	0.33	0.00	0.00	0.00						
5	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50
	2.50	0.33	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20
	2.50	2.50	0.33	0.00						
6	0.33	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20
	2.50	2.50	0.33	0.00	0.00	0.00	0.00	0.75	0.75	1.20
	1.20	1.20	2.50	2.50						

SCHEME 27A :CROP PATTERN 2 : MAMBAL

127

6

Gross crop water requirements (m^3/s)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongons										
1	0.096	0.096	0.050	0.076	0.034	0.059	0.030	0.000	0.000	0.004
	0.037	0.054	0.056	0.045	0.022	0.000	0.104	0.104	0.046	0.071
	0.048	0.074	0.047	0.000						
2	0.000	0.096	0.086	0.050	0.061	0.034	0.060	0.030	0.000	0.000
	0.021	0.037	0.056	0.056	0.043	0.022	0.000	0.000	0.000	0.000
	0.000	0.026	0.000	0.026						
3	0.000	0.000	0.086	0.086	0.036	0.061	0.034	0.060	0.044	0.000
	0.000	0.097	0.101	0.039	0.063	0.036	0.065	0.036	0.000	0.000
	0.000	0.000	0.000	0.000						
4	0.000	0.000	0.000	0.086	0.073	0.036	0.061	0.034	0.074	0.044
	0.000	0.000	0.000	0.101	0.103	0.037	0.067	0.040	0.069	0.040
	0.000	0.000	0.000	0.000						

5	0.000	0.000	0.000	0.000	0.073	0.073	0.036	0.061	0.048	0.074
	0.054	0.000	0.000	0.000	0.000	0.103	0.104	0.041	0.071	0.044
	0.074	0.044	0.000	0.000						
6	0.000	0.000	0.000	0.000	0.000	0.073	0.076	0.036	0.076	0.048
	0.083	0.054	0.000	0.000	0.000	0.000	0.000	0.104	0.104	0.046
	0.076	0.048	0.077	0.047						

Yield response factors (k)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongon										
1	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33	0.00	1.87
	1.87	3.25	1.80	1.80	1.00	0.00	0.75	0.75	1.20	1.20
	1.20	2.50	2.50	0.33						
2	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33	0.00
	1.87	1.87	3.25	1.80	1.80	1.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00						
3	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33
	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33	0.00
	0.00	0.00	0.00	0.00						
4	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50
	0.33	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50
	0.33	0.00	0.00	0.00						
5	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50
	2.50	0.33	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20
	2.50	2.50	0.33	0.00						
6	0.33	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20
	2.50	2.50	0.33	0.00	0.00	0.00	0.00	0.75	0.75	1.20
	1.20	1.20	2.50	2.50						

SCHEME 45 :CROP PATTERN 4 : MERTAGANGGA

45

2

Gross crop water requirements (m^3/s)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongons										
1	0.227	0.000	0.336	0.336	0.139	0.239	0.132	0.232	0.171	0.000
	0.000	0.082	0.159	0.218	0.206	0.167	0.104	0.000	0.406	0.406
	0.279	0.379	0.283	0.383						
2	0.000	0.000	0.000	0.336	0.286	0.139	0.240	0.132	0.287	0.171
	0.000	0.000	0.395	0.395	0.227	0.327	0.236	0.336	0.236	0.000
	0.000	0.000	0.000	0.000						

Yield response factor (k)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongon										
1	2.50	0.33	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33
	0.00	1.87	1.87	3.25	1.80	1.80	1.00	0.00	0.75	0.75
	1.20	1.20	1.20	2.50						
2	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50
	0.33	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33
	0.00	0.00	0.00	0.00						

SCHEME 50 :CROP PATTERN 3 :PERAUPAN

50
4
Gross crop water requirements (m^3/s)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongons										
1	0.000	0.103	0.092	0.054	0.038	0.064	0.036	0.060	0.000	0.000
	0.000	0.023	0.044	0.060	0.057	0.046	0.029	0.000	0.000	0.000
	0.000	0.000	0.000	0.000						
2	0.000	0.000	0.092	0.092	0.038	0.038	0.064	0.036	0.075	0.000
	0.000	0.000	0.109	0.109	0.040	0.068	0.042	0.070	0.043	0.000
	0.000	0.000	0.000	0.000						
3	0.000	0.000	0.000	0.092	0.079	0.038	0.038	0.064	0.051	0.075
	0.000	0.000	0.000	0.000	0.110	0.110	0.044	0.072	0.047	0.074
	0.047	0.000	0.000	0.000						
4	0.000	0.000	0.000	0.000	0.079	0.079	0.038	0.038	0.079	0.051
	0.085	0.000	0.000	0.000	0.000	0.000	0.112	0.112	0.049	0.077
	0.052	0.080	0.050	0.000						

Yield response factor (k)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongon										
1	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33	0.00
	0.00	1.87	1.87	3.25	1.80	1.80	1.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00						
2	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33
	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33
	0.00	0.00	0.00	0.00						
3	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50
	0.33	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50
	2.50	0.33	0.00	0.00						
4	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50
	2.50	0.33	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20
	1.20	2.50	2.50	0.33						

SCHEME 65 :CROP PATTERN 3 : OONGAN
65
4
Gross crop water requirements (m^3/s)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongons										
1	0.000	0.836	0.750	0.436	0.310	0.517	0.295	0.487	0.000	0.000
	0.000	0.184	0.355	0.486	0.461	0.374	0.232	0.000	0.000	0.000
	0.000	0.000	0.000	0.000						
2	0.000	0.000	0.750	0.750	0.310	0.310	0.519	0.295	0.606	0.000
	0.000	0.000	0.884	0.884	0.326	0.550	0.345	0.569	0.346	0.000
	0.000	0.000	0.000	0.000						
3	0.000	0.000	0.000	0.750	0.639	0.310	0.311	0.519	0.417	0.606
	0.000	0.000	0.000	0.000	0.895	0.895	0.361	0.585	0.380	0.604
	0.384	0.000	0.000	0.000						
4	0.000	0.000	0.000	0.000	0.639	0.639	0.311	0.311	0.641	0.417
	0.690	0.000	0.000	0.000	0.000	0.000	0.908	0.908	0.398	0.622
	0.423	0.647	0.410	0.000						

Yield response factor (k)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongon										
1	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33	0.00
	0.00	1.87	1.87	3.25	1.80	1.80	1.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00						
2	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33
	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33
	0.00	0.00	0.00	0.00						
3	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50
	0.33	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50
	2.50	0.33	0.00	0.00						
4	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50
	2.50	0.33	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20
	1.20	2.50	2.50	0.33						

SCHEME 71 :CROP PATTERN 3 : BATAN NYUH
71
4
Gross crop water requirements (m^3/s)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongons										
1	0.000	0.207	0.186	0.108	0.077	0.128	0.073	0.121	0.000	0.000
	0.000	0.046	0.088	0.120	0.114	0.093	0.057	0.000	0.000	0.000
	0.000	0.000	0.000	0.000						
2	0.000	0.000	0.186	0.186	0.077	0.077	0.129	0.073	0.150	0.000
	0.000	0.000	0.219	0.219	0.081	0.136	0.085	0.141	0.086	0.000
	0.000	0.000	0.000	0.000						
3	0.000	0.000	0.000	0.186	0.158	0.077	0.077	0.129	0.103	0.150
	0.000	0.000	0.000	0.000	0.222	0.222	0.089	0.145	0.094	0.150
	0.095	0.000	0.000	0.000						
4	0.000	0.000	0.000	0.000	0.158	0.158	0.077	0.077	0.159	0.103
	0.171	0.000	0.000	0.000	0.000	0.000	0.225	0.225	0.098	0.154
	0.105	0.160	0.101	0.000						

Yield response factor (k)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongon										
1	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33	0.00
	0.00	1.87	1.87	3.25	1.80	1.80	1.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00						
2	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33
	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33
	0.00	0.00	0.00	0.00						
3	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50
	0.33	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50
	2.50	0.33	0.00	0.00						
4	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50
	2.50	0.33	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20
	1.20	2.50	2.50	0.33						

SCHEME 77 :CROP PATTERN 3 : BADUNG
77
4
Gross crop water requirements (m^3/s)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongons										
1	0.000	0.237	0.212	0.123	0.088	0.146	0.084	0.138	0.000	0.000
	0.000	0.052	0.100	0.138	0.131	0.106	0.066	0.000	0.000	0.000
	0.000	0.000	0.000	0.000						
2	0.000	0.000	0.212	0.212	0.088	0.088	0.147	0.084	0.172	0.000
	0.000	0.000	0.250	0.250	0.092	0.156	0.098	0.161	0.098	0.000
	0.000	0.000	0.000	0.000						
3	0.000	0.000	0.000	0.212	0.181	0.088	0.088	0.147	0.118	0.172
	0.000	0.000	0.000	0.000	0.254	0.254	0.102	0.166	0.108	0.171
	0.109	0.000	0.000	0.000						
4	0.000	0.000	0.000	0.000	0.181	0.181	0.088	0.088	0.182	0.118
	0.196	0.000	0.000	0.000	0.000	0.000	0.257	0.257	0.113	0.176
	0.120	0.183	0.116	0.000						

Yield response factor (k)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongon										
1	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33	0.00
	0.00	1.87	1.87	3.25	1.80	1.80	1.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00						
2	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33
	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33
	0.00	0.00	0.00	0.00						
3	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50
	0.33	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50
	2.50	0.33	0.00	0.00						
4	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50
	2.50	0.33	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20
	1.20	2.50	2.50	0.33						

SCHEME 81:CROP PATTERN 3 : MERGAYA
81
4
Gross crop water requirements (m^3/s)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongons										
1	0.000	0.207	0.186	0.108	0.077	0.128	0.073	0.121	0.000	0.000
	0.000	0.046	0.088	0.120	0.114	0.093	0.057	0.000	0.000	0.000
	0.000	0.000	0.000	0.000						
2	0.000	0.000	0.186	0.186	0.077	0.077	0.129	0.073	0.150	0.000
	0.000	0.000	0.219	0.219	0.081	0.136	0.085	0.141	0.086	0.000
	0.000	0.000	0.000	0.000						
3	0.000	0.000	0.000	0.186	0.158	0.077	0.077	0.129	0.103	0.150
	0.000	0.000	0.000	0.000	0.222	0.222	0.089	0.145	0.094	0.150
	0.095	0.000	0.000	0.000						

4	0.000	0.000	0.000	0.000	0.158	0.158	0.077	0.077	0.159	0.103
	0.171	0.000	0.000	0.000	0.000	0.000	0.225	0.225	0.098	0.154
	0.105	0.160	0.101	0.000						

Yield response factor (k)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongon										
1	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33	0.00
	0.00	1.87	1.87	3.25	1.80	1.80	1.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00						
2	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33
	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33
	0.00	0.00	0.00	0.00						
3	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50
	0.33	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50
	2.50	0.33	0.00	0.00						
4	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50
	2.50	0.33	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20
	1.20	2.50	2.50	0.33						

SCHEME 88 CROP PATTERN 3 : MATI
88
4

Gross crop water requirements (m^3/s)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongons										
1	0.000	0.142	0.127	0.074	0.053	0.088	0.050	0.083	0.000	0.000
	0.000	0.031	0.060	0.082	0.078	0.063	0.039	0.000	0.000	0.000
	0.000	0.000	0.000	0.000						
2	0.000	0.000	0.127	0.127	0.053	0.053	0.088	0.050	0.103	0.000
	0.000	0.000	0.150	0.150	0.055	0.093	0.058	0.096	0.059	0.000
	0.000	0.000	0.000	0.000						
3	0.000	0.000	0.000	0.127	0.108	0.053	0.053	0.088	0.071	0.103
	0.000	0.000	0.000	0.000	0.152	0.152	0.061	0.099	0.064	0.102
	0.065	0.000	0.000	0.000						
4	0.000	0.000	0.000	0.000	0.108	0.108	0.053	0.053	0.109	0.071
	0.117	0.000	0.000	0.000	0.000	0.000	0.154	0.154	0.067	0.105
	0.072	0.110	0.069	0.000						

Yield response factor (k)

	Nov Apr Sept	Nov Apr Sept	Dec May Oct	Dec May Oct	Jan June	Jan June	Feb July	Feb July	Mar Aug	Mar Aug
Golongon										
1	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33	0.00
	0.00	1.87	1.87	3.25	1.80	1.80	1.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00						
2	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33
	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50	0.33
	0.00	0.00	0.00	0.00						
3	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50	2.50
	0.33	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50
	2.50	0.33	0.00	0.00						

4	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20	1.20	2.50
	2.50	0.33	0.00	0.00	0.00	0.00	0.75	0.75	1.20	1.20
	1.20	2.50	2.50	0.33						

APPENDIX C SOURCE CODE FOR WATER ALLOCATION PROBLEM

This appendix contains the source code for the GA used to solve the Tukad Ayung system with the reduced gene length.

Table C-1 Contents of the files used in the GA model

File name	**Description**
GA43spcl.c	Name of source code file
Projcal.lst	Name of input project file
ay-cal.nta	Name of network file
ay-gl-03.dma	Name of demand file
ay-2.ifa	Name of inflow file
flagbali.dat	Name of file to specified sink nodes
sinkNcl.out	Sink node output file
supNcl.out	Supply node output file
reach8.out	Reach flow output file
solu_D.out	Input data file for initialise initial population
ga43Ncl.out	output file for overall detail

//GA43spcl.c GA for Tukad Ayung Irrigation system

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <process.h>
#include <time.h>

#define POPSIZE   100    //input population size even number only
#define PXOVER    0.95   //input crossover probability
#define LENGTH    25     //input length of a chromosome
#define genMAX    500    //input maximum generation
#define Pmutate   0.03   //input mutation probability
#define INTERVAL  300    //input interval of generation to recheck of improvement fitnness
#define DELTA     0.001  //input minimum improvement of fitness after INTERVAL generation
#define NREACH    69     //input number of reach
#define NNODE     56     //input number of node
#define NSTEP     24     //input number of time step
#define NSINK     10     //input number of sink node
#define NSCHEME   13     //input number of scheme

void initial(int,float);
float RandReal (float, float);
int  RandInt (int, int);
void Read_inputfiles();
void SwapIntValues(int *, int *);
void SwapRealValues(float *, float *);
void nrerror(char *messg) ;
void objective(int[],int,int,int,int[]);
void sumnum();
void reproduct_matepop();
void SelectMate();
void UniSelectMate();
void mutatepop(int,int);

int   check1[POPSIZE], check2[POPSIZE], oversup[POPSIZE] ;
int   arno[NREACH], nno[NNODE], index, nreach, mondex ;
int   cal_indx[NNODE][NNODE], pofi, C_ind[NREACH];
int   qindx[NNODE][NREACH], inrno[NNODE][NNODE], mated[2*POPSIZE];
int   nnosnk[NNODE], scheme_node[NSCHEME];

float  upper[NSTEP][LENGTH], lower[NSTEP][LENGTH], sum_inflo[NSTEP], ftnss[POPSIZE+1];
float  num_reproduct[POPSIZE], qmx[NREACH],qcap[NNODE][NNODE],
         maxq[LENGTH], sumd[NSTEP] ;
float mated_pop[POPSIZE][LENGTH],GENE[LENGTH][POPSIZE+1],inflo[NSTEP][NNODE],
         dem[NSTEP][NNODE] ;
float  tempq[NSTEP][NREACH], temps[NSTEP][NNODE], tempx[NSTEP][NNODE] ;
float  Delta_ftnss, Old_minftnss, Modi_muta;
float  bestgene[LENGTH][1], bestsink, pGENE[LENGTH];

int   pgen;
float  bestfit, psumsupply;
float  pratio[NNODE],pconstraint1[NNODE], pexcess[NNODE], pnodeftnss[NNODE],
pBalnc[NNODE], pQConect[NNODE],pQsink[NNODE];
float  pofx[NNODE],pdiffer[NNODE],pQ[NREACH],pXEquity[NNODE];
```

```
FILE   *f1,*f2,*f3,*f4,*f5,*f6,*f7,*f8;

//MAIN PROGRAM
void main (void)
{
   int   jj, kk, k, flag[NNODE], gen, nyrs, nchk, iyear, iyear2, round;
   int   ifyr, ifyr2, nno3[NNODE], tt;
   int   i, j, nflow[LENGTH], indice, Tstep_l, Tstep_u;
   int   ii, ic2, ic3, t2, t3, i2, i3;
   float  down,up, TotalTime ;
   char   title3[80];

   clock_t start, finish;
   float   duration;

         if(!(f1 = fopen("ga43Ncl.out","w"))) {
                  printf("can't open output file\n");
                  exit(1);
         }

         if(!(f4 = fopen("flagbali.dat","r"))) {
                  printf("can't open flagbali.dat\n");
                  exit(1);
         }

   for (k=0; k< NNODE; k++){
                  fscanf(f4,"%d",&flag[k]);
   }
   (void) fclose(f4);

   Read_inputfiles();

         if(!(f2 = fopen("Ay-2.ifa","r"))) {
         nrerror("can't open inflows file");
                  exit(1);
         }
   if(!(f3 = fopen("solu_D.out","r"))) {
                  printf("can't open solucal.out\n");
                  exit(1);
         }
   if(!(f5 = fopen("soluNcl.out","w"))) {
                  printf("can't open solu_E3.out\n");
                  exit(1);
         }
         if(!(f6 = fopen("sinkNcl.out","w"))) {
                  printf("can't open sinkD.out\n");
                  exit(1);
         }
         if(!(f7 = fopen("reachNcl.out","w"))) {
                  printf("can't open reachD.out\n");
                  exit(1);
         }
         if(!(f8 = fopen("supNcl.out","w"))) {
                  printf("can't open supplyD.out\n");
                  exit(1);
         }
   fprintf(f1,"**********************************\n");
   fprintf(f1,"**GA Program for Water Allocation**\n");
```

```
fprintf(f1,"**********************************\n");
fprintf(f1,"POPSIZE %5d LENGTH %5d\n", POPSIZE,   LENGTH );

        fgets(title3, 80, f2);
        fscanf(f2," %d ", &nyrs);
        fscanf(f2," %d ", &nchk);

if(nchk != NNODE) nrerror(" problem with no of nodes in INFLOWS file.") ;

TotalTime = 0.0f;

        for(kk = 0; kk < nyrs; kk++) {
                fscanf(f2,"%d", &iyear);
                if(kk == 0) ifyr = iyear;
                if(iyear != ifyr+kk) nrerror("problem with iyear#1");
        for(jj = 0; jj < NNODE; jj++) {
        fscanf(f2," %d ", &nno3[jj]) ;
                }
                fscanf(f2,"\n");

                fscanf(f3,"%4d",&iyear2);
                if(kk == 0) ifyr2 = iyear2;
          if(iyear2 != ifyr2+kk) nrerror("problem with iyear#2");

   fprintf(f5,"%4d\n",iyear);

   for (jj=0; jj<LENGTH; jj++){
                   fscanf(f3," %d ", &nflow[jj]);
                   fprintf(f5,"%5d",nflow[jj]);
                }
   fprintf(f5,"\n");
                fscanf(f3,"\n");

   fprintf(f6," %d\n", iyear) ;  fprintf(f6,"       ") ;
                for(ii = 0; ii < NSTEP; ii++) fprintf(f6,"%7d",ii) ;
                fprintf(f6,"\n") ;

   fprintf(f7," %d\n", iyear) ;  fprintf(f7,"       ") ;
                for(ii = 0; ii < NSTEP; ii++) fprintf(f7,"%7d",ii) ;
                fprintf(f7,"\n") ;

   fprintf(f8," %d\n", iyear) ;  fprintf(f8,"       ") ;
                for(ii = 0; ii < NSTEP; ii++) fprintf(f8,"%7d",ii) ;
                fprintf(f8,"\n") ;

// Start of Bimonthly Loop

          for(tt = 0; tt<NSTEP; tt++){
                index  = 0;
                indice = 0;
                   sum_inflo[tt] = 0.0;
                         for(jj = 0; jj < NNODE; jj++){
                                  fscanf(f2," %f ", &inflo[tt][jj]);
                                  sum_inflo[tt] = sum_inflo[tt] + inflo[tt][jj];
                         }
                         fscanf(f2,"\n");
```

```
        fscanf(f3,"%d",&Tstep_l) ;
        for (j=0; j<LENGTH; j++){
         fscanf(f3,"%f",&lower[tt][j]);
                if(lower[tt][j] < 0.00f) lower[tt][j] = 0.0f;
        }
        fscanf(f3,"\n") ;

        fscanf(f3,"%d",&Tstep_u) ;
        if(Tstep_u != Tstep_l) nrerror("problem with Tstep_setting limit file ");
        for (j=0; j<LENGTH; j++){
         fscanf(f3,"%f",&upper[tt][j]);
                if(upper[tt][j] < 0.0f) upper[tt][j] = (-1)*upper[tt][j];
        }
        fscanf(f3,"\n") ;

        for(j=0; j<LENGTH; j++)  {
                if (lower[tt][j] > upper[tt][j]){
                        SwapRealValues(&lower[tt][j],&upper[tt][j]);
                }
        }

      srand(1);
      start = clock();
      initial(tt,0.0001f);
      Modi_muta     = 0.15f;
      ftnss[0]  = 0.0f;
      ftnss[POPSIZE]  = 100000.0f;
      Old_minftnss   = 100000.0f;
      Delta_ftnss    = 100000.0f;
      bestsink      = 1000000.0f;
      bestfit  = 1000000.0f;

        do{
                  mondex = 0;
        for(gen = 0; gen <= genMAX; gen++) {
                objective(flag, gen, tt, iyear, nflow);
                if( index == 999) break;
                sumnum();
                reproduct_matepop();
                UniSelectMate();
                mutatepop(tt,gen);

  if (!((gen+1)%INTERVAL)){
        Delta_ftnss = (float)fabs((double)((Old_minftnss - ftnss[POPSIZE])/ftnss[POPSIZE]));
        Old_minftnss = ftnss[POPSIZE];
  }
}//end of Generation loop
Modi_muta = Modi_muta/10;

round = RAND_MAX;
       for(i=0;i<POPSIZE;i++){
         for(j=0;j<LENGTH;j++){
           if(mondex == 777) GENE[j][i] = (float)((bestgene[j][0] -
0.01f)+0.0001f*(float)floor((1+10000.0*((bestgene[j][0] + 0.01f)-(bestgene[j][0] -
0.01f)))*(float)rand()/(round+1.00)));
                        else GENE[j][i] = (float)((GENE[j][POPSIZE] -
0.01f)+0.0001f*(float)floor((1+10000.0*((GENE[j][POPSIZE] + 0.01f)-(GENE[j][POPSIZE] -
0.01f)))*(float)rand()/(round+1.00)));
```

```
                                }
                            }

        } while (index != 999);

         finish = clock();
    duration = (float)(finish - start) / CLOCKS_PER_SEC;
         TotalTime = (float)(TotalTime + duration);
    printf(" Time = %5.2f sec\n", duration);
    fprintf(f1,"Time = %5.2f sec \n\n", duration);

        //this is to print best gene for lower and upper boundaries
        for(i = 0; i<1; i++){
          fprintf(f5,"%2d",tt+1);

          for(jj = 0; jj < LENGTH; jj++){
                down = (float)(GENE[jj][POPSIZE] - 0.02);
                if(down < 0.0f) down = 0.0f;
                fprintf(f5,"%7.3f",down);
          }
          fprintf(f5,"\n");

          fprintf(f5,"%2d",tt+1);
          for(jj = 0;jj < LENGTH;jj++){
            up   = (float)(GENE[jj][POPSIZE] + 0.02);
            if(jj == 0) up = inflo[tt][0];
            if(up > maxq[jj]) up = (float)(maxq[jj]);
            fprintf(f5,"%7.3f",up);
          }
          fprintf(f5,"\n");
        }
} //end of Bimonthly(Timestep [tt]) loop

    for(i3 = 0; i3< NSCHEME; i3++) {
                ic3 = scheme_node[i3] ;
                    fprintf(f8," X%3d       ", ic3) ;
                        for (t3=0; t3 < NSTEP; t3++){
          fprintf(f8,"%7.3f", tempx[t3][i3]);
                                }
                    fprintf(f8,"\n");
                }
    fprintf(f8,"\n");

    for(i2 = 0; i2<NSINK; i2++) {
                ic2 = nnosnk[i2] ;
                    fprintf(f6," S%2d       ", ic2) ;
                        for (t2=0; t2 < NSTEP; t2++){
           fprintf(f6,"%7.3f", temps[t2][i2]);
                                }
                    fprintf(f6,"\n");
                }

                for (i1=0; i1<NREACH; i1++){
        ic1 = arno[i1] ;
                        fprintf(f7," Q%3d       ", ic1) ;
          for (t1=0; t1 < NSTEP; t1++){
                fprintf(f7,"%7.3f", tempq[t1][i1]) ;
                                }
```

```
                    fprintf(f7,"\n");
                }
        fprintf(f7,"\n");

    fprintf(f1,"End of Year >>>>>>\n\n");
    fprintf(f1,"Sum_time in this year = %5.2f min.", TotalTime/60);
    printf("Sum_time in this year = %5.2f min.\n", TotalTime/60);

        }// end of Annual loop

 fprintf(f1,"Total Time Consumed = %7.2f hours\n",TotalTime/3600);
 printf("Total Time Consumed = %7.2f hours\n",TotalTime/3600);
 printf("End of run  ENJOY!!!! ");

 (void) fclose(f1);
 (void) fclose(f3);

} // end of MAIN program

/**********START OF SUBROUTINES **********/
void Read_inputfiles()
{
        int  ii, jj, ir, id, nk, nnodes, nsink;
        int          num_schemes, rno[NNODE][NNODE], ip, ip1, ip2, ip3;
        int  node_num, num_gol[NSCHEME], igol, is, it;
        int  kk, rr, iv;
        float tdem, k_val[NSTEP][80], dem_gol[NSTEP][80], sumdemgol[NSTEP][NSCHEME];
        char  title[80], title2[220];
        char  networkfile[20], demandsfile[20];

        if(!(f2 = fopen("Projcal.lst","r"))) {
   nrerror(" Can not open project file") ;
  }
  // now read the data file names
        fgets(networkfile, 20, f2) ;
        fgets(demandsfile, 20, f2) ;

        for(ii = 0; ii < 20; ii++) {
                if(networkfile[ii] == '\n') networkfile[ii] = '\0' ;
                if(demandsfile[ii] == '\n') demandsfile[ii] = '\0' ;
        }
        (void) fclose(f2) ;

  /**********************/
  // Now read network data

        if(!(f2 = fopen(networkfile,"r"))) {
   nrerror(" Unable to open network file") ;
  }

  fgets(title, 80, f2);
        fscanf(f2," %d", &nnodes) ;

        tdem = 0 ;
        for(jj = 0; jj < nnodes; jj++){
                fscanf(f2," %d ", &nno[jj]) ;
        }
        fscanf(f2,"\n");
```

```
        fscanf(f2,"%d ", &nsink) ;
        for(jj = 0; jj < nsink; jj++) {
                fscanf(f2,"%d ", &nnosnk[jj]) ;
        }
fscanf(f2,"\n");

fscanf(f2," %d ",&num_schemes);
        for(jj=0;jj<num_schemes;jj++) {
                fscanf(f2,"%d ", &scheme_node[jj]);
        }
fscanf(f2,"\n");

fgets(title, 80, f2);       // Connection Index
        for(jj = 0; jj < nnodes; jj++) {
                fscanf(f2,"%d", &nk) ;
                if(nk != nno[jj]) {
            nrerror("Hello: Index array error") ;
        }
        }
        fscanf(f2,"\n") ;

        for(jj = 0; jj<nnodes; jj++) {
                qindx[jj][jj] = 0 ;
                fscanf(f2,"%3d ", &nk) ;
                if(nk != nno[jj]) {
                  nrerror("Index array error 2") ;
                }
                for(ii = 0; ii<nnodes; ii++) {
                        fscanf(f2,"%2d", &qindx[jj][ii]) ;
                }
   fscanf(f2,"\n") ;
        }
fgets(title, 80, f2);       // Reach Capacities

        for(jj = 0; jj < nnodes; jj++) {
                fscanf(f2,"%3d ", &nk) ;

                if(nk != nno[jj]) {
            nrerror("array error 3") ;
        }
        }
        fscanf(f2,"\n") ;

        for(jj = 0; jj < nnodes; jj++) {
                qcap[jj][jj] = 0.0;
                fscanf(f2,"%3d ", &nk) ;

                if(nk != nno[jj]) {
             nrerror(" array error 1") ;
        }
                for(ii = 0; ii<nnodes; ii++) {
                        fscanf(f2,"%f",&qcap[jj][ii]) ;

                }
                fscanf(f2,"\n") ;
        }

fgets(title,80,f2);         // Reach Numbers
```

```
        for(jj = 0; jj < nnodes; jj++) {
                fscanf(f2,"%3d ", &nk) ;
                if(nk != nno[jj]) {
                  nrerror(" array error#2") ;
                }
        }
        fscanf(f2,"\n") ;

        for(jj = 0; jj < nnodes; jj++) {
                fscanf(f2,"%d", &nk) ;
                if(nk != nno[jj]) {
                  nrerror(" array error#3") ;
                }

        for(ii = 0; ii < nnodes; ii++) {
                        fscanf(f2," %d ", &rno[jj][ii]) ;
                }
        }
        fgets(title, 80, f2);       // Reach Calculation Index

                for(jj = 0; jj < nnodes; jj++) {
                fscanf(f2,"%3d", &nk) ;
                if(nk != nno[jj]) {
            nrerror("Hello: Reach Calculation Index array error") ;
        }
        }

        fscanf(f2,"\n") ;
        for(jj = 0; jj<nnodes; jj++) {
                cal_indx[jj][jj] = 0 ;
                fscanf(f2,"%3d", &nk) ;
                if(nk != nno[jj]) {
                  nrerror("Index array error 2") ;
                }

                for(ii = 0; ii<nnodes; ii++) {
                        fscanf(f2,"%3d", &cal_indx[jj][ii]) ;
                }
                fscanf(f2,"\n") ;
        }
    id = 0 ;
          for(jj = 0; jj < nnodes; jj++) {
                  for(ii = jj+1; ii <= nnodes; ii++) {
                        if(qindx[jj][ii] != 0){
                                C_ind[id] = cal_indx[jj][ii] ;
                                   id++ ;
                                }
                  }
                }

(void)fclose(f2) ;

        // Determine the number of reaches, and assign internal nos

        nreach = 0 ;
        for(jj = 0; jj < nnodes; jj++) {
                inrno[jj][jj] = 0 ;
                for(ii = jj+1; ii < nnodes; ii++) {
```

```
                        if(qindx[jj][ii] == 0 && rno[jj][ii] != 0)
                                nrerror("1#Index and reach no.arrays out of sinc. ") ;
                        if(qindx[jj][ii] == 0 && qcap[jj][ii] != 0)
                                nrerror("2#Index and capacity arrays out of sinc. ") ;
                        if(qindx[jj][ii] != 0){
                                inrno[jj][ii] = nreach ;
                                nreach++ ;
                        }
                        inrno[ii][jj] = inrno[jj][ii] ;
                }
        }
        // check reach numbers

        nreach = 0;
        for(jj = 0; jj < nnodes; jj++) {
                for(ii = jj+1; ii <= nnodes; ii++) {
                        if(qindx[jj][ii] != 0){
                                arno[nreach] = rno[jj][ii] ;
                                nreach++ ;
                        }
          }
        }

        ir = 0 ;
        for(jj = 0; jj < nnodes; jj++) {
                for(ii = jj+1; ii <= nnodes; ii++) {
                        if(qindx[jj][ii] != 0){
                                qmx[ir] = qcap[ii][jj] ;
                                ir++ ;
                        }
                }
        }
// Need to read the demands in here !!

        if(!(f2 = fopen(demandsfile, "r"))) {
    nrerror(" can't open demands file") ;
  }

        ip1=0;
        ip2 = ip3 = 0 ;
        jj = 0 ;
        rr = 0;
        for(kk = 0; kk < nnodes; kk++) {
                if(nno[kk] == scheme_node[jj]) {

                        fgets(title2, 220, f2);         // Scheme: CROP PATTERN : KADEWATAN

                fscanf(f2,"%d", &node_num);      // 6
                if(node_num != scheme_node[jj]) nrerror(" scheme demands out of sinc") ;

                fscanf(f2,"%d\n", &num_gol[jj]); // 4
                fgets(title2, 220, f2);         // Gross crop water requirements
                fgets(title2, 220, f2);                 // Nov Nov Dec .....Oct
                fgets(title2, 220, f2);                 // Golongon

                        for(igol = 0; igol < num_gol[jj]; igol++) {
                        fscanf(f2,"%d ", &is);       // 1 2 3 4 ...
```

```
                                for(ii = 0; ii < NSTEP; ii++) {

                                    fscanf(f2,"%f ", &dem_gol[ii][ip1]) ;
                                }
                        ip1++ ;
                }
            ip3++ ;

                fgets(title2, 220, f2);     /* this should be crop coefficients */

                fgets(title2, 220, f2);
                fgets(title2, 220, f2);

                for(igol = 0; igol < num_gol[jj]; igol++) {
                fscanf(f2,"%d ", &is) ;
                                for(ii = 0; ii < NSTEP; ii++)  {
                                        fscanf(f2,"%f ", &k_val[ii][ip2]) ;
                                }
                        ip2++ ;
                }
                fscanf(f2,"\n") ;
        jj++ ;

                } // end of if scheme_nodes

}       // end of nnodes

                        jj = 0;
                        iv = 0;
                        for(kk = 0; kk < nnodes; kk++) {
                                if(nno[kk] == scheme_node[jj]) {
                                        for(it = 0; it < NSTEP; it++) {
                                                ip=iv;
                                                sumdemgol[it][jj]  = 0.0f;
                                          for(igol = 0; igol < num_gol[jj]; igol++) {

            sumdemgol[it][jj] = sumdemgol[it][jj] + dem_gol[it][ip];
            ip++;
                                                        }

                                        }
                                        jj++;
                                        iv=ip;
                                }
                        }
/*************************/
    jj = 0;
    for(kk = 0; kk < NNODE; kk++) {
                        if(nno[kk] == scheme_node[jj]) {
                          for(it = 0; it < NSTEP; it++) {
        dem[it][kk] = sumdemgol[it][jj];
                                }
        jj++;
                        }
                        else {
                                for(it =0; it < NSTEP; it++) {
                                  dem[it][kk] = 0.0f;
                                }
```

```
                    }
                }

        for(it = 0; it < NSTEP; it++) {
                        sumd[it] = 0.0f;
            for(kk = 0; kk < NNODE; kk++) {
                                sumd[it] = sumd[it] + dem[it][kk];
                    }
                }

   (void)fclose(f2) ;

} // close demand file
/*******************/
void initial(int tt,float inc)
{
  int i,j,mate;
  mate = RAND_MAX;

  for(j=0;j<LENGTH;j++) {
          bestgene[j][0] = (float)(lower[tt][j]+inc*(float)floor((1+(upper[tt][j]-
lower[tt][j])/inc)*(float)rand()/(mate+1.00)));
  }
  fprintf(f1,"\n");

  for(i=0;i<POPSIZE;i++){
    for(j=0;j<LENGTH;j++){
    GENE[j][i] = (float)(lower[tt][j]+inc*(float)floor((1+(upper[tt][j]-
lower[tt][j])/inc)*(float)rand()/(mate+1.00)));
  }
  }
}
/*************************/
float RandReal (float lower,float upper)
{
          float val;
          float nb = ( upper - lower );
          val =  (float)(rand()%1000/1000.0*nb + lower);
          return ((float)val);
}
/*************************/
int RandInt (int lower,int upper)
{
          int val;
          int nb =.(upper - lower );
          val =  (rand()%nb + lower);
          return val;
}
/*************************/
void  objective(int flag[],int gen,int tt, int iyear, int nflow[])
{
  int   i,j,k,m,hh,ii,jj,jr,kk,kt,mm,i2,i3,seton;
  float  sumQnreach,sumQout[NNODE],sumsupply[POPSIZE], sum_sink[POPSIZE],
Qflowinto[POPSIZE][NNODE];
  float  sumftnss, stringftnss,avgftnss,minftnss;
  float  pselct[POPSIZE], sumpselct, avgpselct, minpselct;
  float  pp[POPSIZE][NNODE], QConect[POPSIZE][NNODE], Q[NREACH][POPSIZE];
  float  Qsink[POPSIZE][NNODE], x[POPSIZE][NNODE], global_bal[POPSIZE] ;
```

```
float  Balnc[POPSIZE][NNODE], nodeftnss[POPSIZE][NNODE], excess[POPSIZE][NNODE] ;
float  ratio[POPSIZE][NNODE], constraint1[POPSIZE][NNODE];
float  sumconstr1[POPSIZE], sumconstr2[POPSIZE], sumconstr3, sumconstr4[POPSIZE] ;
float  overflow[POPSIZE][NREACH];

for(i=0; i<POPSIZE; i++){
                k = 0;
                for(j=0;j<NREACH;j++){
                        if (C_ind[j] == 0 && qmx[j] >= 0.0f) {
                  Q[j][i] = GENE[k][i];
                  k++;
                        }
                        else {
                  Q[j][i] = 0.0f;
                        }
                }
}

kk=0;
for(m=0; m<NREACH; m++){
          if (qmx[m] >= 0.0f && C_ind[m] == 0) {
             maxq[kk] = qmx[m];
                kk++;
          }
}

 sumftnss = 0.0f;
        sumconstr3 = 0.0f;
for(i=0;i<POPSIZE;i++)  {
 kt=0;
        sumconstr1[i] = 0.0f;
 sumconstr2[i] = 0.0f;
 sumconstr4[i] = 0.0f;
        sumsupply[i]  = 0.0f;
        stringftnss   = 0.0f;
        sum_sink[i]   = 0.0f;
        check1[i] = 0;
        check2[i] = 0;
          for(jj = 0; jj <NNODE; jj++){
          QConect[i][jj]= 0.0f;
          Qflowinto[i][jj] = 0.0f;
          sumQnreach = 0.0;
          sumQout[jj]= 0.0f;
           for(ii=0;ii <NNODE; ii++) {
                    if(qindx[jj][ii]!=0){
                      nreach = inrno[ii][jj];
                            if( qcap[ii][jj] < 0.0f && qindx[jj][ii] == -1) {
                                    Q[nreach][i] = (-1)*qcap[ii][jj]*sumQnreach;
                            }
                            else if(cal_indx[jj][ii] == 0 && qindx[jj][ii] == -1) {
          sumQout[jj] = sumQout[jj] + Q[nreach][i];
                            }
                            else if(cal_indx[jj][ii] == 1 && qindx[jj][ii] == -1) {
                               Q[nreach][i] = (float)fabs(inflo[tt][jj] + sumQnreach - sumQout[jj]);
                            }
                            else if( qcap[ii][jj] >= 0.0f || qindx[jj][ii] == +1) {
                                    sumQnreach  = sumQnreach + Q[nreach][i];
                            }
```

```
                        QConect[i][jj] = QConect[i][jj]+qindx[jj][ii]*Q[nreach][i];

        if(QConect[i][jj] > 0.0f){
                                Qflowinto[i][jj] = Qflowinto[i][jj] + QConect[i][jj];
                        }
                }
        }

        if (flag[jj] == 1) Qsink[i][jj] = QConect[i][jj];
  if (flag[jj] == 0) Qsink[i][jj] = 0.0f;
        sum_sink[i] = (float)(sum_sink[i] + Qsink[i][jj]);

        if   (dem[tt][jj] == 0.0f)  x[i][jj] = 0.0f;
        else  x[i][jj] = (float)(inflo[tt][jj] + QConect[i][jj] - Qsink[i][jj]);

        if(x[i][jj] > 0.0f) sumsupply[i] =  sumsupply[i] + x[i][jj];

        //This is nodal balance constraint
  Balnc[i][jj] = (float)fabs((double)(inflo[tt][jj] + QConect[i][jj] - x[i][jj] - Qsink[i][jj]));

        if(Balnc[i][jj] > 0.001f){
                if(inflo[tt][jj] + Qflowinto[i][jj] > 0.0f) {
                        constraint1[i][jj]  =
(float)(pow(Balnc[i][jj],2))/(inflo[tt][jj]+Qflowinto[i][jj]) ;
                }
                else constraint1[i][jj] = 0.0f;
        }
        sumconstr1[i]  =  sumconstr1[i] + constraint1[i][jj];

        if(dem[tt][jj] != 0.0f)  ratio[i][jj] = (float)(x[i][jj]/dem[tt][jj]);
        else ratio[i][jj] = 0.0f;

        //This is supply constraint
        if(x[i][jj] > dem[tt][jj])  excess[i][jj] = (x[i][jj] - dem[tt][jj])/dem[tt][jj];
        else excess[i][jj] = 0.0f;
        sumconstr2[i]  =  sumconstr2[i] + excess[i][jj];

        // This is equitable function
        pp[i][jj] = (float)fabs((double)(dem[tt][jj]-x[i][jj]));

        if (dem[tt][jj] != 0.0) nodeftnss[i][jj] = (float)(1/dem[tt][jj])*((float)pow(pp[i][jj],2));
  else nodeftnss[i][jj] = 0.0f;
        stringftnss = (float)(stringftnss + nodeftnss[i][jj]);

    if (Balnc[i][jj] < 0.001f) check1[i]++;
                if (x[i][jj] - dem[tt][jj] <= 0.001) check2[i]++;

  }//end of jj

                // This is global water balance
          global_bal[i] =  (float)( sum_inflo[tt] - sumsupply[i] - sum_sink[i])/sum_inflo[tt]);

    sumconstr3  =  sumconstr3 + global_bal[i];

                // This is capacity constraints
    for(m=0; m<NREACH; m++){
          if (qmx[m] >= 0.0f) {
                        if(Q[m][i] > qmx[m]) {
```

```
                                overflow[i][m] = (Q[m][i] - qmx[m])/qmx[m];
                        }
                        else overflow[i][m] = 0.0f;
        sumconstr4[i] = sumconstr4[i] + overflow[i][m];
            kt++;
                }
               }

  //This is objective function
  ftnss[i] = (float)( stringftnss + sumconstr1[i] + sumconstr2[i] + sumconstr3 + sumconstr4[i]);
  sumftnss = sumftnss + ftnss[i];

      //This to choose the best fitness
        if(bestfit > ftnss[i]){
                mondex = 777;
                bestfit  = ftnss[i];

                if(bestsink > sum_sink[i]){
                        bestsink = sum_sink[i];
                }

                pgen    = gen;
                pofi    = i;
    psumsupply  = sumsupply[i];

                for(j = 0; j <LENGTH; j++){
                        bestgene[j][0] = GENE[j][POPSIZE];
                }

    for(jj=0; jj<NNODE; jj++){
                  pratio[jj]      = ratio[i][jj];
                        pconstraint1[jj] = constraint1[i][jj];
                        pexcess[jj]      = excess[i][jj];
                        pnodeftnss[jj]   = nodeftnss[i][jj];
                        pBalnc[jj]       = Balnc[i][jj];
                        pQConect[jj]     = QConect[i][jj];
                        pQsink[jj]       = Qsink[i][jj];
                        pofx[jj]         = x[i][jj];
                }
    for(hh=0; hh<NREACH; hh++) pQ[hh] = Q[hh][i];
                for(mm=0; mm<LENGTH; mm++) pGENE[mm] = GENE[mm][i];

               }


} //end of i loop

      // This is to terminate after no improvement after INTERVAL generation
      if(Delta_ftnss <= DELTA){

            index = 999;
                                psumsupply = sum_inflo[tt] - bestsink;
                                fprintf(f1,"%4d T%2d",iyear,tt);

                        fprintf(f1," %7.4f %5d %2d\n", bestfit,check1[pofi],check2[pofi]);

                                fprintf(f1,"inf=%7.3f",sum_inflo[tt]);
       fprintf(f1,"sup=%7.3f",psumsupply);
```

```
                fprintf(f1,"dem=%7.3f",sumd[tt]);
        fprintf(f1,"sink  =%7.3f\n",bestsink);

                                printf("%4d T%2d",iyear,tt);

                        printf("%7.4f %5d %2d\n", bestfit,check1[pofi],check2[pofi]);

                                printf("inf=%7.3f",sum_inflo[tt]);
        printf("sup=%7.3f",psumsupply);
                        printf("dem=%7.3f",sumd[tt]);
        printf("sink=%7.3f",bestsink);


                                // print sink output to file f6
                                for( jj = 0; jj < NNODE; jj++) {
                                        for(i2 = 0; i2<NSINK; i2++){

                                                if( nno[jj] == nnosnk[i2]) {
            temps[tt][i2] = pQsink[jj];
                                                }
                                }
                                }

        // print reach file output to f7
        for(jr = 0; jr < NREACH; jr++) {

                                tempq[tt][jr] = pQ[jr];
                                }

                                // print supply output to file f8
        for( jj = 0; jj < NNODE; jj++) {
                                        for(i3 = 0; i3<NSCHEME; i3++){

                                                if( nno[jj] ==  scheme_node[i3]){
            tempx[tt][i3] = pofx[jj];
                                                }
                                }
                                }
 }//end of if loop

avgftnss = sumftnss/POPSIZE;
sumpselct = 0.0;
for(i=0; i<POPSIZE; i++)  {
   pselct[i] = ftnss[i]/sumftnss;
        sumpselct = sumpselct + pselct[i];
}
avgpselct = sumpselct/POPSIZE;

minpselct = pselct[0];
minftnss  = ftnss[0];
seton = POPSIZE;
  for(i=0; i< POPSIZE; i++) {
     if(ftnss[i]  < minftnss)  minftnss  = ftnss[i];
          if(pselct[i] < minpselct) minpselct = pselct[i];
                if(ftnss[i]  < ftnss[POPSIZE]){
                  ftnss[POPSIZE] = ftnss[i];
                  seton = i;
                }
```

```
            }

    for(j=0; j<LENGTH; j++){
                GENE[j][POPSIZE] = GENE[j][seton];
            }

 for(i=0;i<= POPSIZE;i++) {
     if(pselct[i] == minpselct)       num_reproduct[i] = 2;
     else if(pselct[i] >  avgpselct/2)  num_reproduct[i] = 0;
     else if(pselct[i] <= avgpselct/2)  num_reproduct[i] = 1;
     else num_reproduct[i] = 0;
 }
//printf("%3d %7.2f %7.2f %4d %2d\n",gen,ftnss[POPSIZE],bestsink,check1[pofi],check2[pofi]);
//fprintf(f1,"%3d %7.2f %7.2f\n",gen,ftnss[POPSIZE],bestsink);

}
/*********************/
void sumnum()
{
 int  a,i,mate,mate1[POPSIZE];
 float sumnum_repro;

 sumnum_repro = 0;
 for(i=0;i<POPSIZE;i++) {
          sumnum_repro = sumnum_repro + num_reproduct[i];
 }
 sumnum_repro+=2;


 mate = RAND_MAX;
 mate1[0] = (int)floor(sumnum_repro *(float)rand()/mate);
 for(i=1;i<POPSIZE;i++) {
          mate1[i] = mate1[i-1] + 1;
          if (mate1[i] >= sumnum_repro)
                   mate1[i] = 0;
 }
 for(i=0; i<POPSIZE; i+=2){
          mated[i]   = mate1[i/2];
          mated[i+1] = mate1[i/2 + POPSIZE/2]; //half deck of card
 }

 a = (int)floor(POPSIZE *(float)rand()/mate); //shuffle
 for(i=0; i<POPSIZE; i++){
          if(a>=POPSIZE) a =0;
          mate1[i] = mated[a];
          a++;
 }

 for(i=0; i<POPSIZE; i+=2){  //half deck of card
          mated[i] = mate1[i/2];
          mated[i+1] = mate1[i/2 + POPSIZE/2];
 }

}
/*********************/
void  reproduct_matepop()
{
 int  m,ii,k1, i,j,R;
```

```
  float repro_pop[2*POPSIZE][LENGTH];

  for(j=0;j<LENGTH;j++) {
            repro_pop[0][j] = bestgene[j][0];
            repro_pop[1][j] = bestgene[j][0];
  }

  ii=2;
  for(m=0;m<POPSIZE ;m++) {
             for(k1=0; k1<num_reproduct[m];k1++) {
          for(j=0;j<LENGTH;j++) {
            repro_pop[ii][j] = GENE[j][m];
                                  }
     ii++;
             }
  }

  for (i=0;i<POPSIZE ;i++) {
    R = mated[i];
    for (j=0;j<LENGTH;j++)  {
      mated_pop[i][j] = repro_pop[R][j];
    }
  }

}
/*********************/
void XOVER(int i1, int i2)
{
          int j,kk;

          kk = RandInt(0,LENGTH-1);

          for(j=kk;j<LENGTH;j++)  {
        SwapRealValues(&mated_pop[i1][j],&mated_pop[i2][j]);
   }
}
/*********************/
void UNIXOVER(int i1, int i2)
{
          int j,kk;

          for(j=0;j<LENGTH;j++)  {
                    kk= rand()&01;
      if(kk==1) SwapRealValues(&mated_pop[i1][j],&mated_pop[i2][j]);
   }
}
/*********************/
void TPXOVER(int i1, int i2)
{
          int j,kk1,kk2;

          kk1 = RandInt(0,LENGTH-1);
          kk2 = RandInt(0,LENGTH-1); // this is for 2-site XOVER
          if (kk1>kk2) SwapIntValues(&kk1,&kk2); // this is to swap 2-site if the later is less

          for(j=kk1;j<kk2;j++)  {
        SwapRealValues(&mated_pop[i1][j],&mated_pop[i2][j]);
   }
```

```
}
/*********************/
void SelectMate()
{
        int  mem, num_select,one;
   float Prob_X;

   num_select = 0;
        for (mem = 0; mem< POPSIZE ; mem++) {
                Prob_X = rand()%1000/1000.0f;
                if (Prob_X < PXOVER){
                        ++num_select;
                        if(num_select%2 == 0){
                                XOVER(one,mem);
                        }
                        else one = mem;
                }
        }

}
/*********************/
void UniSelectMate()
{
        int  mem, num_select,one;
   float Prob_X;

   num_select = 0;
        for (mem = 0; mem< POPSIZE ; mem++) {
                Prob_X = rand()%1000/1000.0f;
                if (Prob_X < PXOVER){
                        ++num_select;
                        if(num_select%2 == 0){
                                UNIXOVER(one,mem);
                        }
                        else one = mem;
                }
        }

}
/*********************/
void mutatepop(int tt, int gen)
{
 int i,j,mutant;
 double r;

        for (i=0;i<POPSIZE;i++) {

                for (j=0;j<LENGTH;j++) {

                        //this is nonuniform mutation
        r = rand()%1000/1000.0;
                        if(r<Pmutate){
        mutant = rand()&01;
        if(gen < genMAX*0.1){
        if (mutant==1) mated_pop[i][j] = mated_pop[i][j] + (float)Modi_muta;
                else mated_pop[i][j] = mated_pop[i][j] - (float)Modi_muta;
        }
        else if((gen >= genMAX*0.1) && (gen < genMAX*0.2)) {
```

```
            if (mutant==1) mated_pop[i][j] = mated_pop[i][j] + (float)Modi_muta/2;
                    else mated_pop[i][j] = mated_pop[i][j] - (float)Modi_muta/2;
            else if((gen >= genMAX*0.2) && (gen < genMAX*0.3)) {
            if (mutant==1) mated_pop[i][j] = mated_pop[i][j] + (float)Modi_muta/3;
                    else  mated_pop[i][j] = mated_pop[i][j] - (float)Modi_muta/3;
            }
            else if((gen >= genMAX*0.3) && (gen < genMAX*0.4)) {
            if (mutant==1) mated_pop[i][j] = mated_pop[i][j] + (float)Modi_muta/4;
                    else mated_pop[i][j] = mated_pop[i][j] - (float)Modi_muta/4;
            }
            else if((gen >= genMAX*0.4) && (gen < genMAX*0.5)) {
             if (mutant==1) mated_pop[i][j] = mated_pop[i][j] + (float)Modi_muta/5;
                    else mated_pop[i][j] = mated_pop[i][j] - (float)Modi_muta/5;
            }
            else if((gen >= genMAX*0.5) && (gen < genMAX*0.6)) {
            if (mutant==1) mated_pop[i][j] = mated_pop[i][j] + (float)Modi_muta/6;
                    else mated_pop[i][j] = mated_pop[i][j] - (float)Modi_muta/6;
            }
            else if((gen >= genMAX*0.6) && (gen < genMAX*0.7)) {
             if (mutant==1) mated_pop[i][j] = mated_pop[i][j] + (float)Modi_muta/7;
                    else mated_pop[i][j] = mated_pop[i][j] - (float)Modi_muta/7;
            }
            else if((gen >= genMAX*0.7) && (gen < genMAX*0.8)) {
            if (mutant==1) mated_pop[i][j] = mated_pop[i][j] + (float)Modi_muta/8;
                    else mated_pop[i][j] = mated_pop[i][j] - (float)Modi_muta/8;
            }
            else if((gen >= genMAX*0.8) && (gen < genMAX*0.9)) {
             if (mutant==1) mated_pop[i][j] = mated_pop[i][j] + (float)Modi_muta/9;
                    else mated_pop[i][j] = mated_pop[i][j] - (float)Modi_muta/9;
            }
             else {    if (mutant==1) mated_pop[i][j] = mated_pop[i][j] + (float)Modi_muta/10;
                    else mated_pop[i][j] = mated_pop[i][j] - (float)Modi_muta/10;
             }
            if ( mated_pop[i][j] < 0.0f)    mated_pop[i][j] = 0.0f;
                    else if ( mated_pop[i][j] > maxq[j]) mated_pop[i][j] = maxq[j];
            }
            GENE[j][i] = mated_pop[i][j];

            }//end of j
//end of i
}
/*********************/
void nrerror(char *messg)
{
        puts(messg) ;
   exit(1) ;
}
/*********************/
void SwapIntValues(int *xx, int *yy)
{
        int temp;
        temp = *xx;
        *xx=*yy;
        *yy=temp;
}
/*********************/
```

```
void SwapRealValues(float *xx, float *yy)
{
        float temp;
        temp = *xx;
        *xx=*yy;
        *yy=temp;
}
/*********************/
```

APPENDIX D METHOD USED FOR WATER SCHEDULING

Soil Water Content

Gravimetric soil water content(θ_m) is expressed in terms of the mass of water in a unit mass of soil (Warrick 1990). Volumetric water content(θ_v) is more commonly used and is expressed in terms of the volume of water in a unit volume of soil. As irrigation amounts are normally expressed on a volume basis and the water stored is often considered on a root-zone basis, volumetric water content is the preferred term. Soil moisture content near the wilting point is not readily available to the plant, thus the term readily available soil moisture has been used to refer to that portion of available moisture that is easily extracted by plants. The objective of irrigation is to keep the available moisture for plant within the zone of easy extraction.

Soil water has been classified as hygroscopic, capillary, and gravitational (Hansen et al. 1980). The soil water availability may be considered in the zones as shown in Figure D-1

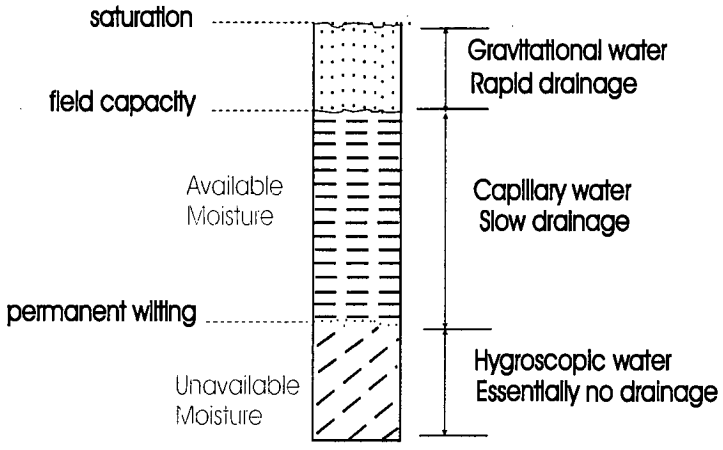

Figure D.1 *Soil water availability to plant and drainage characteristics*
(after Hansen et al. 1980)

At saturation, the pores of the soil are filled with water. With free drainage, water drains until the soil is holding the maximum quantity of water that can be retained against the gravity. Soil in this state is said to be at field capacity (Bailey 1990).

Field Capacity

Field capacity has been commonly assumed to be the upper limit of soil water available to the crops. Ratliff et al.(1983) defined field capacity as the soil water content that the daily rate of drainage was reduced to between 0.1 and 0.2% of saturated moisture content, following a thorough wetting of the soil. An excellent summary of the historical and the present dynamic concepts of field capacity is presented by Ahuja and Nielsen(1990). Early studies by Alway and McDole (1917); Israelson and West (1922); and Veihmeyer and Hendrickson (1927, 1931) showed that, following wetting, changes in soil water content decrease rapidly with time. The internal drainage rate generally becomes negligibly small or even zero in just a few days. The soil moisture content at which internal drainage supposedly stopped was termed the field capacity. Field capacity has been widely accepted as a physical characteristic of soil. A widely used laboratory method for the estimation of field capacity has been the soil water content in equilibrium within a suction of 10 kPa. Field measurements with tensiometers indicate that in most soils, the soil water suction sustained after 2-3 days drainage is within the range of 5 to 10 kPa (100 kPa = 1.0 bar)(Bruce et al. 1983; Nofziger et al.1983). For irrigations, it has been commonly accepted that the application of a certain quantity of water to the soil should only fill the soil moisture deficit to field capacity. The amount of irrigation to be applied is determined at any time on the basis of deficit from the field capacity in the root zone to be wetted. The objective of irrigation is to bring the soil moisture back to field capacity.

In the dynamic approach to soil moisture modeling developed by Richards et al.(1956); Ogata and Richards (1957), field capacity is no longer considered a constant soil property. The internal drainage of soil is continual and shows neither sharp changes nor constant levels. In most soils, internal drainage can persist at an appreciable rate for many days except for deep coarse-textured soils. The drainage rate and time at which it becomes negligible depends on many factors including depth to the groundwater table, evaporation from soil surface and the uptake of water by plants (Ahuja and Nielsen 1990), as well as the physical characteristics of soil layers.

Soil Water Availability

Concepts of soil water availability to plants are excellently summarized by Ahuja and Nielsen (1990). The classical view, first conceived in the 1920s, was that soil moisture is equally available to most crops in their respective rooting zones throughout a definable range of soil wetness, from an upper limit called field capacity to a lower limit called permanent wilting point. These were considered to be characteristic and constant values for

each soil and independent of crop or climate (Veihmeyer and Hendrickson 1927; 1949; 1950; 1955). The old concept of equal availability of water to plants between field capacity and permanent wilting point often resulted in large, infrequent irrigations (Ahuja and Nielsen 1990). Water was applied when available water was nearly depleted. The amount applied equaled that needed to fill the current root zone to field capacity, plus a leaching fraction. Using this concept the crop could possibly be subjected to extreme conditions of poor aeration and excessive drying (Rawlins and Raats 1975; Hillel 1980).

In modern concepts of irrigation management, the field is perceived to be a unified system named the Soil-Plant-Atmosphere Continuum(SPAC), in which all processes are interdependent. Although soil moisture is considered to be available in the range between field capacity and permanent wilting point, it is not equally available. The soil moisture is not a property of soil alone but indeed is a function of the plant, soil, and climate. This dynamic concept of soil water availability has significant implications for irrigation management. The concept requires more flexible irrigation application, based on meteorological conditions, plant growth stage, root system, and soil water flow rates. The rate of water uptake depends on the ability of the roots to absorb water from the soil with which they are in contact. It also depends on the ability of the soil to supply and transmit water toward the roots at a rate sufficient to meet transpiration and growth requirements (Gardner 1960; Huck and Hillel 1983).

Allen et al.(1998) suggested an equation to calculate total available soil water or so called maximum available water content which can be stored within the root zone, and which can be used by the crop as follows:

$$TAW = 1000 \cdot (FC - PWP) \cdot Zr \qquad \text{(D-1)}$$

where: FC = the moisture content at field capacity ($m^3\ m^{-3}$)

PWP = the moisture content at permanent wilting point ($m^3\ m^{-3}$)

Zr = the effective depth of root zone (m)

TAW = the total available soil water in the root zone (mm)

The allowable depletion of soil water is the equivalent depth of water which can be safely depleted from the root zone between irrigations without the crop becoming stressed. It is also called usable available water or readily available water (*RAW*) in some soil moisture documents. The allowable depletion fraction is dependent upon the fraction of the soil water available to crop roots. It varies as the crop develops. The readily available moisture defines water available to the crop and can be computed as follows (Rijov et al. 1973):

$$RAW = \sum_{i=1}^{n} (FC_i - PWP_i).d_i.f_i \qquad \text{(D-2)}$$

where: RAW = the readily available soil water in the root zone which can be depleted before the crop is adversely affected (mm)

n = the number of soil layers or increments sampled within the effective rooting depth (mm)

i = layer i-th of soil

d_i = the depth of soil layer in the i-th layers

f_i = the ratio of allowable depletion to the total available water in the i-th layers

Allowable Depletion Fraction

In irrigation scheduling or crop modeling, the ratio of actual to maximal transpiration (the relative transpiration rate) is assumed to decrease linearly when the soil dries out below a critical available water value (Brisson 1998). This critical available value, as indicated above, is usually expressed as a fraction F of TAW. The factors which influence F are the crop, the soil, the climate, and the management conditions (Rijov et al. 1973). As crops are growing, the effective soil depth increases. This in turn increases the maximum available water content but may not be just proportional to the rooting depth because in the deeper soil layers the root density may not be sufficient to take up all of the available water (Brisson 1998). Brisson (1998) studied an analytical solution for the estimation of the critical available soil water fraction for a single layer water balance model and growing crops. The aim of the study was to use the basic laws governing water transfer through the plants at a daily time step to compute F dynamically as the crop was growing. It was found that F decreased as the root system grows in depth, and that differences in the shape of root profile can be responsible for different water stress sensitivity in the early stage of growth. Conversely, F is relatively insensitive to the average root radius. F is most sensitive to atmospheric demand and rooting depth. Burch et al. (1978) noticed a significant difference in the root profiles of sorghum and soybean on water uptake. It was found that at the completing stage of root growth, the F value seems to be controlled by the taproot depth not the shape of the root profile. This conclusion agrees with the results of the Ehlers et al. (1991). The work of Brisson was based on a single soil layer.

Since information for each soil depth within the root zone is often lacking, the following simplified expression as suggested in FAO Irrigation and drainage paper 56 (Allen et al. 1998) may be used:

$$RAW = 1000 \cdot (FC - PWP) \cdot Zr \cdot F \quad \text{(D-3)}$$

or $$RAW = TAW \cdot F$$

where; F = average fraction of TAW that can be depleted from the root zone before moisture stress ($F \leq 1$)

The value of F normally varies from 0.30 for shallow rooted plants at high evapotranspiration rates (> 8 mm/day) to 0.7 for deep rooted plants at low transpiration rate(< 3 mm/day) (Allen et al. 1998). The value of F can be adjusted for different ETc according the following equation. However, the adjusted F is limited at the range of $0.1 \leq F \leq 0.8$.

$$F = F_{FAO\_Table\ 22} + 0.04 \cdot (5 - ETc) \quad \text{(D-4)}$$

where; ETc = crop evapotranspiration under standard conditions* (mm/day)

$F_{FAO\_Table\ 22}$ = page 2 chapter 8, Irrigation and drainage paper 56

*ETc under standard condition means the evapotranspiration from disease-free, well-fertilised crops, grown in large fields, under optimum soil water conditions and achieving full production under the given climate conditions.

Crop Evapotranspiration

Crop evapotranspiration (ETc) under standard condition has been presented by Allen et al. (1998) as follows:

$$ETc = Kc \cdot ETo \quad \text{(D-5)}$$

where; ETo = crop reference evapotranspiration by FAO Penman-Monteith method (mm/day)

Kc = crop coefficient (dimensionless)

Crop Coefficient

The crop coefficient integrates the effects of characteristics that distinguish the cropped surface from a reference grass surface. There are two approaches in calculating the crop coefficient:- the single crop coefficient, and dual crop coefficient (Allen et al. 1998). In the single crop coefficient, the difference in evapotranspiration between the cropped and reference grass is combined into one single coefficient. As soil evaporation may fluctuate daily as a result of rainfall or irrigation, the single crop coefficient express only the time-

averaged crop coefficient. The approach is often used to compute *ETc* for weekly or longer time periods.

In the dual crop coefficient approach, the effects of crop transpiration and soil evaporation are determined separately. The basal crop coefficient (*Kcb*) describes crop transpiration, and is defined as the ratio of *ETc* over *ETo* when the soil surface layer is dry in the time that the average soil water content of the root zone is adequate to sustain full plant transpiration. *Kcb* represents the coefficient when there is not the additional effects of soil wetting by irrigation or precipitation. *Kcb* values are provided in three stages of crop growth (Allen et al. 1998); initial stage (Kcb_{ini}), middle growth stage (Kcb_{mid}) and harvest stage (Kcb_{end}).

The soil water evaporation coefficient (*Ke*) describes evaporation from the soil surface. If the soil surface becomes drier, the *Ke* becomes smaller, and the *Ke* falls to zero when there is no water left for evaporation. The dual crop coefficient is presented as:

$$Kc = Kcb + Ke \qquad \text{(D-6)}$$

where; Kcb = basal crop coefficient

Ke = soil water evaporation coefficient

The selection of the *Kc* approach depends on the purpose of the calculation, the accuracy required, the climate data and the time step with which the calculation are executed. In real time irrigation scheduling, the dual crop coefficient approach is suggested. The *ETc* with dual crop coefficient for irrigation scheduling is presented as:

$$ETc = (Kcb + Ke) \cdot ETo \qquad \text{(D-7)}$$

This equation is use when relative humidity (RH) > 45%, and the mean value for daily wind speed (U_2) at 2 m height over grass during the mid or late season growth stage is in the range of 1 m/s $\leq U_2 \leq$ 6 m/s. Outside this range the adjustment is required. Full discussion of *Kcb* have been provided by Allen et al. (1998) for various crop types in the condition of non-stress, well-managed crops in subhumid climates for use with the FAO Penman-Montieth *ETo*.

Crop Evapotranspiration Under Soil Water Stress Conditions

If the soil water drops below wilting point value, the crops said to be water stressed. The effects of soil water stress are described by Allen et al. (1998) as follows;

$$ETadj = (Ks \cdot Kcb + Ke) \cdot ETo \qquad \text{(D-8)}$$

where; $ETadj$ = crop evapotranspiration adjust for soil water stress condition

Ks = water stress coefficient

Ks is a dimensionless transpiration reduction factor. For soil water limiting conditions, $Ks < 1$. Where there is no soil water stress, $Ks = 1$. The Ks could be estimated as:

$$Ks = \frac{TAW - \theta_r}{TAW - RAW} \quad \text{(D-9)}$$

or

$$Ks = \frac{TAW - \theta_r}{(1 - F) \cdot TAW} \quad \text{(D-10)}$$

where; θ_r = residual soil moisture content in root zone (mm)

The Effective Rooting Depth

The effective root zone which is calculated on each day (Zr_i) could be estimated as (Allan et al. 1998):

$$Zr_i = Zr_{min} + (Zr_{max} - Zr_{min}) \cdot \frac{Kcb_i - Kcb_{ini}}{Kcb_{mid} - Kcb_{ini}} \text{ for } J < J_{mid} \quad \text{(D-11)}$$

and

$$Zr_i = Zr_{max} = Zr_{max} \text{ for } J \geq J_{mid} \quad \text{(D-12)}$$

where; Zr_i = effective depth of the root zone on day i (m)

Zr_{min} = initial effective depth of the root zone at the beginning of the initial period of planting (m)

Zr_{max} = maximum effective depth of the root zone during the midseason period (m) (from Table 22 in the appendix)

Kcb_i = Kcb at the day in question

Kcb_{ini} = Kcb at initial stage of growth

Kcb_{mid} = Kcb at the mid growth stage

J = day of year (1 to 365)

J_{mid} = day of year at the midseason period of growth stage

For annual crops, Zr_i should be presented as the depth of seed placement plus an additional depth of soil that may contribute water to the seed as it extends its initial roots downward following germination. For many annual crops, Zr_{min} can be estimated as 0.15 to 0.20 m (Allen et al. 1998).

Soil Water Balance Techniques

Irrigation scheduling usually deals with "when" and "how much" to irrigate. The study of Singh et al.(1995) described the soil water balance equation to determine irrigation required at the end of time *t* as:

$$I_t = ETc_t - P_t \pm R_{o_t} \pm D_{c_t} \pm \theta_r \qquad \text{(D-13)}$$

where: I_t = irrigation requirement (mm);

ETc_t = crop evapotranspiration (mm);

P_t = precipitation (mm);

R_{o_t} = surface runoff into (+) or out of (-) the field in question (mm);

D_{c_t} = capillary drainage toward the surface (-) or into the subsurface (+) (mm);

θ_r = residual soil moisture content in root zone (mm)

Where the surface is flat such, as in most irrigation settings, surface runoff ($\pm R_o$) is negligible (Curwen and Massie 1985). Similarly where the ground water table is well beneath the rooting depth of the soil, capillary rise ($-D_c$) is also negligible. Downward percolation ($+D_c$) is also negligible where heavy soils restrict downward drainage of soil moisture. Irrigation requirement then could be rewritten as:

$$I_t = ETc_t - P_t \pm \theta_r \qquad \text{(D-14)}$$

P is routinely measured and *ETc* can be estimated as shown earlier. The crop starts to wilt when soil moisture is at wilting point and the crop dies when soil moisture is at permanent wilting point. Thus, θ_r must be kept above permanent wilting point even in the water stress condition. This study set the critical value for θ_r as at start of wilting point, i.e θ_r is measured relative to wilting point. The wilting point addressed in this study is defined as the soil moisture fraction at the point that the crops start to wilt. The permanent wilting point is defined as the soil moisture fraction at which permanent wilting of the plant leaf has occurred and applying additional water will not relieve the wilted condition.

Since rooting depth of plant changes with crop growth stage, soil moisture content of the following time period (θ_t) can be calculated as follows:

$$\theta_t = \frac{\theta_{t-1} \cdot Zr_{t-1} + \theta_p \cdot (Zr_t - Zr_{t-1})}{Zr_t} \qquad \text{(D-15)}$$

where: θ_{t-1} = soil moisture content of the preceding growth stage *t* (mm)

θ_t = soil moisture content of this growth stage *t* (mm)

θ_p = initial soil moisture content of total profile (mm)

Zr_{t-1} = rooting depth of plant of the preceding growth stage(m)

Zr_t = rooting depth of plant of this growth stage(m)

In calculation by the equation (D-15), it is assumed in the condition that the soil moisture content will stay same when the time past but in fact it is dynamics.

APPENDIX E DATA FILES FOR WATER SCHEDULING TEST SYSTEMS

This appendix contains the input data file for water scheduling test systems. Table E-1 presents the data input and its description for the simple system. In the test of stress and non-stress conditions, canal capacities are changed as outlined in chapter 6.

Table E-2 described input data for the more complex system, for both zero-1 and warabandi approaches. Canal capacity in main, secondary and tertiary canals are changed as outlined in chapter 7.

Table E-1. Data file input for test system (chapter 6)

Data file and Comments

Number of scheme

3

Scheme number

1 2 3

Scheme area (ha)

100 100 100

Number staggering (days) from the start

0 0 3

Main canal full supply (m^3/sec.)

0.24 (*for stress condition used 0.14)

Secondary canal full supply (m^3/sec.)

0.12 0.12 0.12 (*for stress condition used 0.07)

Crop requirement in each scheme (mm/day)

Stage number

1 2 3 4 5 6 7 8 9 10 11 12

13 14 15 16 17 18 19 20 21 22 23

24 25 26 27 28 29 30 31 32 33 34

35 36 37 38 39 40 41 42 43 44 45

46 47 48 49 50 51 52 53 54 55 56

57 58 59 60 61 62 63 64 65 66 67

68 69 70 71 72 73 74 75 76 77 78

79 80 81 82 83 84 85 86 87 88 89

90 91 92 93 94 95 96 97 98 99

100

Stage day

1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1

	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1

CWR(mm)

0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.74	0.74	0.74	0.74
	0.74	0.74	0.74	0.74	0.74	0.73	0.73	0.73	0.73	0.73
	0.73	0.73	0.72	0.72	0.91	1.10	1.29	1.47	1.66	1.85
	2.03	2.21	2.40	2.58	2.76	2.94	3.12	3.30	3.48	3.65
	3.83	4.01	4.18	4.35	4.53	4.70	4.87	5.04	5.21	5.20
	5.18	5.17	5.16	5.15	5.13	5.12	5.11	5.10	5.09	5.07
	5.06	5.05	5.04	5.02	5.01	5.00	4.99	4.97	4.96	4.95
	4.94	4.93	4.91	4.90	4.89	4.88	4.86	4.85	4.84	4.64
	4.44	4.24	4.04	3.85	3.65	3.46	3.26	3.07	2.88	2.69
	2.50	2.31	2.12	1.93	1.75	1.56	1.38	1.19	1.01	
0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.74	0.74	0.74	0.74
	0.74	0.74	0.74	0.74	0.74	0.73	0.73	0.73	0.73	0.73
	0.73	0.73	0.72	0.72	0.91	1.10	1.29	1.47	1.66	1.85
	2.03	2.21	2.40	2.58	2.76	2.94	3.12	3.30	3.48	3.65
	3.83	4.01	4.18	4.35	4.53	4.70	4.87	5.04	5.21	5.20
	5.18	5.17	5.16	5.15	5.13	5.12	5.11	5.10	5.09	5.07
	5.06	5.05	5.04	5.02	5.01	5.00	4.99	4.97	4.96	4.95
	4.94	4.93	4.91	4.90	4.89	4.88	4.86	4.85	4.84	4.64
	4.44	4.24	4.04	3.85	3.65	3.46	3.26	3.07	2.88	2.69
	2.50	2.31	2.12	1.93	1.75	1.56	1.38	1.19	1.01	
0.75	0.75	0.75	0.75	0.74	0.74	0.74	0.74	0.74	0.74	0.74
	0.74	0.74	0.73	0.73	0.73	0.73	0.73	0.73	0.73	0.72
	0.72	0.72	0.72	0.72	0.91	1.09	1.28	1.47	1.65	1.83
	2.02	2.20	2.38	2.56	2.74	2.92	3.10	3.28	3.45	3.63
	3.80	3.98	4.15	4.32	4.49	4.67	4.83	5.00	5.17	5.16
	5.15	5.13	5.12	5.11	5.10	5.09	5.07	5.06	5.05	5.04
	5.02	5.01	5.00	4.99	4.97	4.96	4.95	4.94	4.93	4.91
	4.90	4.89	4.88	4.86	4.85	4.84	4.83	4.82	4.80	4.61

	4.41	4.21	4.01	3.82	3.62	3.43	3.24	3.05	2.86	2.67
	2.48	2.29	2.10	1.92	1.73	1.55	1.37	1.18	1.00	

Effective rainfall (mm/day)

0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	

Field capacity (m^3/m^3)

0.17 0.17 0.17

Permanent wilting point (m^3/m^3)

0.07 0.07 0.07

Length Development Stages (days)

25.0 25.0 30.0 20.0

Basal Crop Coefficient

0.15 1.14 0.25

Rooting depth of crop (m)

0.15 0.80

Allowable depletion fraction

0.45

Table E-2. Data file input for a more complex test system

Data file and Comments				
Canal data file for zero-1 criteria and warabandi approach				
Number of schemes				
4				
Scheme number				
1	2	3	4	
Number of tertiary canal in each scheme				
4	4	4	4	
Number of crop				
1				
1	1	100	Secondary canal No.1,	tertiary 1, crop area (ha)
	2	100		tertiary 2, crop area (ha)
	3	100		tertiary 3, crop area (ha)
	4	100		tertiary 4, crop area (ha)
2	1	100	Secondary canal No.2,	tertiary 1, crop area (ha)
	2	100		tertiary 2, crop area (ha)
	3	100		tertiary 3, crop area (ha)
	4	100		tertiary 4, crop area (ha)
3	1	100	Secondary canal No.3,	tertiary 1, crop area (ha)
	2	100		tertiary 2, crop area (ha)
	3	100		tertiary 3, crop area (ha)
	4	100		tertiary 4, crop area (ha)
4	1	100	Secondary canal No.4,	tertiary 1, crop area (ha)
	2	100		tertiary 2, crop area (ha)
	3	100		tertiary 3, crop area (ha)
	4	100		tertiary 4, crop area (ha)
Main canal full supply (m^3/sec.)				
0.28				
Secondary canal full supply (m^3/sec.)				
0.14	0.14	0.14	0.14	

Secondary No. - tertiary canal full supply delivery (m^3/sec)

1	0.07	0.07	0.07	0.07
2	0.07	0.07	0.07	0.07
3	0.07	0.07	0.07	0.07
4	0.07	0.07	0.07	0.07

Secondary No. - lag-day among tertiaries from the start (days) (input integer number)

1	0	0	5	5
2	0	0	5	5
3	10	10	15	15
4	10	10	15	15

Secondary No. - field capacity(m^3/m^3)

1	0.17	0.17	0.17	0.17
2	0.17	0.17	0.17	0.17
3	0.17	0.17	0.17	0.17
4	0.17	0.17	0.17	0.17

Secondary No. - Permanent wilting point (m^3/m^3)

1	0.07	0.07	0.07	0.07
2	0.07	0.07	0.07	0.07
3	0.07	0.07	0.07	0.07
4	0.07	0.07	0.07	0.07

CWR (ETm) in time period i(mm/period) fit polynomial curve is required

Number of crops

1

Crop calendar (days)

100

Crop number

1

Number of stages

100

Stage number

1	2	3	4	5	6	7	8	9	10	11	12
	13	14	15	16	17	18	19	20	21	22	23
	24	25	26	27	28	29	30	31	32	33	34

	35	36	37	38	39	40	41	42	43	44	45
	46	47	48	49	50	51	52	53	54	55	56
	57	58	59	60	61	62	63	64	65	66	67
	68	69	70	71	72	73	74	75	76	77	78
	79	80	81	82	83	84	85	86	87	88	89
	90	91	92	93	94	95	96	97	98	99	
	100										

Stage day

1	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1

CWR (mm)

0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.74	0.74	0.74	0.74
	0.74	0.74	0.74	0.74	0.74	0.73	0.73	0.73	0.73	0.73
	0.73	0.73	0.72	0.72	0.91	1.10	1.29	1.47	1.66	1.85
	2.03	2.21	2.40	2.58	2.76	2.94	3.12	3.30	3.48	3.65
	3.83	4.01	4.18	4.35	4.53	4.70	4.87	5.04	5.21	5.20
	5.18	5.17	5.16	5.15	5.13	5.12	5.11	5.10	5.09	5.07
	5.06	5.05	5.04	5.02	5.01	5.00	4.99	4.97	4.96	4.95
	4.94	4.93	4.91	4.90	4.89	4.88	4.86	4.85	4.84	4.64
	4.44	4.24	4.04	3.85	3.65	3.46	3.26	3.07	2.88	2.69
	2.50	2.31	2.12	1.93	1.75	1.56	1.38	1.19	1.01	
0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.74	0.74	0.74	0.74
	0.74	0.74	0.74	0.74	0.74	0.73	0.73	0.73	0.73	0.73
	0.73	0.73	0.72	0.72	0.91	1.10	1.29	1.47	1.66	1.85
	2.03	2.21	2.40	2.58	2.76	2.94	3.12	3.30	3.48	3.65

	3.83	4.01	4.18	4.35	4.53	4.70	4.87	5.04	5.21	5.20
	5.18	5.17	5.16	5.15	5.13	5.12	5.11	5.10	5.09	5.07
	5.06	5.05	5.04	5.02	5.01	5.00	4.99	4.97	4.96	4.95
	4.94	4.93	4.91	4.90	4.89	4.88	4.86	4.85	4.84	4.64
	4.44	4.24	4.04	3.85	3.65	3.46	3.26	3.07	2.88	2.69
	2.50	2.31	2.12	1.93	1.75	1.56	1.38	1.19	1.01	
0.75	0.75	0.74	0.74	0.74	0.74	0.74	0.74	0.74	0.74	0.74
	0.73	0.73	0.73	0.73	0.73	0.73	0.73	0.72	0.72	0.72
	0.72	0.72	0.72	0.72	0.90	1.09	1.28	1.46	1.64	1.83
	2.01	2.19	2.37	2.55	2.73	2.91	3.09	3.26	3.44	3.61
	3.79	3.96	4.13	4.30	4.47	4.64	4.81	4.98	5.15	5.13
	5.12	5.11	5.10	5.09	5.07	5.06	5.05	5.04	5.02	5.01
	5.00	4.99	4.97	4.96	4.95	4.94	4.93	4.91	4.90	4.89
	4.88	4.86	4.85	4.84	4.83	4.82	4.80	4.79	4.78	4.58
	4.38	4.19	3.99	3.80	3.61	3.41	3.22	3.03	2.84	2.65
	2.47	2.28	2.09	1.91	1.72	1.54	1.36	1.18	1.00	
0.75	0.75	0.74	0.74	0.74	0.74	0.74	0.74	0.74	0.74	0.74
	0.73	0.73	0.73	0.73	0.73	0.73	0.73	0.72	0.72	0.72
	0.72	0.72	0.72	0.72	0.90	1.09	1.28	1.46	1.64	1.83
	2.01	2.19	2.37	2.55	2.73	2.91	3.09	3.26	3.44	3.61
	3.79	3.96	4.13	4.30	4.47	4.64	4.81	4.98	5.15	5.13
	5.12	5.11	5.10	5.09	5.07	5.06	5.05	5.04	5.02	5.01
	5.00	4.99	4.97	4.96	4.95	4.94	4.93	4.91	4.90	4.89
	4.88	4.86	4.85	4.84	4.83	4.82	4.80	4.79	4.78	4.58
	4.38	4.19	3.99	3.80	3.61	3.41	3.22	3.03	2.84	2.65
	2.47	2.28	2.09	1.91	1.72	1.54	1.36	1.18	1.00	
0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.74	0.74	0.74	0.74
	0.74	0.74	0.74	0.74	0.74	0.73	0.73	0.73	0.73	0.73
	0.73	0.73	0.72	0.72	0.91	1.10	1.29	1.47	1.66	1.85
	2.03	2.21	2.40	2.58	2.76	2.94	3.12	3.30	3.48	3.65
	3.83	4.01	4.18	4.35	4.53	4.70	4.87	5.04	5.21	5.20
	5.18	5.17	5.16	5.15	5.13	5.12	5.11	5.10	5.09	5.07
	5.06	5.05	5.04	5.02	5.01	5.00	4.99	4.97	4.96	4.95
	4.94	4.93	4.91	4.90	4.89	4.88	4.86	4.85	4.84	4.64

	4.44	4.24	4.04	3.85	3.65	3.46	3.26	3.07	2.88	2.69
	2.50	2.31	2.12	1.93	1.75	1.56	1.38	1.19	1.01	
0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.74	0.74	0.74	0.74
	0.74	0.74	0.74	0.74	0.74	0.73	0.73	0.73	0.73	0.73
	0.73	0.73	0.72	0.72	0.91	1.10	1.29	1.47	1.66	1.85
	2.03	2.21	2.40	2.58	2.76	2.94	3.12	3.30	3.48	3.65
	3.83	4.01	4.18	4.35	4.53	4.70	4.87	5.04	5.21	5.20
	5.18	5.17	5.16	5.15	5.13	5.12	5.11	5.10	5.09	5.07
	5.06	5.05	5.04	5.02	5.01	5.00	4.99	4.97	4.96	4.95
	4.94	4.93	4.91	4.90	4.89	4.88	4.86	4.85	4.84	4.64
	4.44	4.24	4.04	3.85	3.65	3.46	3.26	3.07	2.88	2.69
	2.50	2.31	2.12	1.93	1.75	1.56	1.38	1.19	1.01	
0.75	0.75	0.74	0.74	0.74	0.74	0.74	0.74	0.74	0.74	0.74
	0.73	0.73	0.73	0.73	0.73	0.73	0.73	0.72	0.72	0.72
	0.72	0.72	0.72	0.72	0.90	1.09	1.28	1.46	1.64	1.83
	2.01	2.19	2.37	2.55	2.73	2.91	3.09	3.26	3.44	3.61
	3.79	3.96	4.13	4.30	4.47	4.64	4.81	4.98	5.15	5.13
	5.12	5.11	5.10	5.09	5.07	5.06	5.05	5.04	5.02	5.01
	5.00	4.99	4.97	4.96	4.95	4.94	4.93	4.91	4.90	4.89
	4.88	4.86	4.85	4.84	4.83	4.82	4.80	4.79	4.78	4.58
	4.38	4.19	3.99	3.80	3.61	3.41	3.22	3.03	2.84	2.65
	2.47	2.28	2.09	1.91	1.72	1.54	1.36	1.18	1.00	
0.75	0.75	0.74	0.74	0.74	0.74	0.74	0.74	0.74	0.74	0.74
	0.73	0.73	0.73	0.73	0.73	0.73	0.73	0.72	0.72	0.72
	0.72	0.72	0.72	0.72	0.90	1.09	1.28	1.46	1.64	1.83
	2.01	2.19	2.37	2.55	2.73	2.91	3.09	3.26	3.44	3.61
	3.79	3.96	4.13	4.30	4.47	4.64	4.81	4.98	5.15	5.13
	5.12	5.11	5.10	5.09	5.07	5.06	5.05	5.04	5.02	5.01
	5.00	4.99	4.97	4.96	4.95	4.94	4.93	4.91	4.90	4.89
	4.88	4.86	4.85	4.84	4.83	4.82	4.80	4.79	4.78	4.58
	4.38	4.19	3.99	3.80	3.61	3.41	3.22	3.03	2.84	2.65
	2.47	2.28	2.09	1.91	1.72	1.54	1.36	1.18	1.00	
0.74	0.74	0.74	0.74	0.74	0.74	0.73	0.73	0.73	0.73	0.73
	0.73	0.73	0.72	0.72	0.72	0.72	0.72	0.72	0.72	0.71

	0.71	0.71	0.71	0.71	0.89	1.08	1.26	1.44	1.63	1.81
	1.99	2.17	2.35	2.52	2.70	2.88	3.05	3.23	3.40	3.57
	3.74	3.91	4.08	4.25	4.42	4.59	4.76	4.92	5.09	5.07
	5.06	5.05	5.04	5.02	5.01	5.00	4.99	4.97	4.96	4.95
	4.94	4.93	4.91	4.90	4.89	4.88	4.86	4.85	4.84	4.83
	4.82	4.80	4.79	4.78	4.77	4.76	4.74	4.73	4.72	4.53
	4.33	4.14	3.94	3.75	3.56	3.37	3.18	2.99	2.81	2.62
	2.44	2.25	2.07	1.88	1.70	1.52	1.34	1.16	0.99	
0.74	0.74	0.74	0.74	0.74	0.74	0.73	0.73	0.73	0.73	0.73
	0.73	0.73	0.72	0.72	0.72	0.72	0.72	0.72	0.72	0.71
	0.71	0.71	0.71	0.71	0.89	1.08	1.26	1.44	1.63	1.81
	1.99	2.17	2.35	2.52	2.70	2.88	3.05	3.23	3.40	3.57
	3.74	3.91	4.08	4.25	4.42	4.59	4.76	4.92	5.09	5.07
	5.06	5.05	5.04	5.02	5.01	5.00	4.99	4.97	4.96	4.95
	4.94	4.93	4.91	4.90	4.89	4.88	4.86	4.85	4.84	4.83
	4.82	4.80	4.79	4.78	4.77	4.76	4.74	4.73	4.72	4.53
	4.33	4.14	3.94	3.75	3.56	3.37	3.18	2.99	2.81	2.62
	2.44	2.25	2.07	1.88	1.70	1.52	1.34	1.16	0.99	
0.74	0.73	0.73	0.73	0.73	0.73	0.73	0.73	0.72	0.72	0.72
	0.72	0.72	0.72	0.72	0.71	0.71	0.71	0.71	0.71	0.71
	0.71	0.70	0.70	0.70	0.88	1.07	1.25	1.43	1.61	1.79
	1.97	2.14	2.32	2.49	2.67	2.84	3.02	3.19	3.36	3.53
	3.70	3.87	4.04	4.20	4.37	4.53	4.70	4.86	5.02	5.01
	5.00	4.99	4.97	4.96	4.95	4.94	4.93	4.91	4.90	4.89
	4.88	4.86	4.85	4.84	4.83	4.82	4.80	4.79	4.78	4.77
	4.76	4.74	4.73	4.72	4.71	4.70	4.69	4.67	4.66	4.47
	4.28	4.09	3.89	3.71	3.52	3.33	3.14	2.96	2.77	2.59
	2.41	2.22	2.04	1.86	1.68	1.50	1.33	1.15	0.97	
0.74	0.73	0.73	0.73	0.73	0.73	0.73	0.73	0.72	0.72	0.72
	0.72	0.72	0.72	0.72	0.71	0.71	0.71	0.71	0.71	0.71
	0.71	0.70	0.70	0.70	0.88	1.07	1.25	1.43	1.61	1.79
	1.97	2.14	2.32	2.49	2.67	2.84	3.02	3.19	3.36	3.53
	3.70	3.87	4.04	4.20	4.37	4.53	4.70	4.86	5.02	5.01
	5.00	4.99	4.97	4.96	4.95	4.94	4.93	4.91	4.90	4.89

	4.88	4.86	4.85	4.84	4.83	4.82	4.80	4.79	4.78	4.77
	4.76	4.74	4.73	4.72	4.71	4.70	4.69	4.67	4.66	4.47
	4.28	4.09	3.89	3.71	3.52	3.33	3.14	2.96	2.77	2.59
	2.41	2.22	2.04	1.86	1.68	1.50	1.33	1.15	0.97	
0.74	0.74	0.74	0.74	0.74	0.74	0.73	0.73	0.73	0.73	0.73
	0.73	0.73	0.72	0.72	0.72	0.72	0.72	0.72	0.72	0.71
	0.71	0.71	0.71	0.71	0.89	1.08	1.26	1.44	1.63	1.81
	1.99	2.17	2.35	2.52	2.70	2.88	3.05	3.23	3.40	3.57
	3.74	3.91	4.08	4.25	4.42	4.59	4.76	4.92	5.09	5.07
	5.06	5.05	5.04	5.02	5.01	5.00	4.99	4.97	4.96	4.95
	4.94	4.93	4.91	4.90	4.89	4.88	4.86	4.85	4.84	4.83
	4.82	4.80	4.79	4.78	4.77	4.76	4.74	4.73	4.72	4.53
	4.33	4.14	3.94	3.75	3.56	3.37	3.18	2.99	2.81	2.62
	2.44	2.25	2.07	1.88	1.70	1.52	1.34	1.16	0.99	
0.74	0.74	0.74	0.74	0.74	0.74	0.73	0.73	0.73	0.73	0.73
	0.73	0.73	0.72	0.72	0.72	0.72	0.72	0.72	0.72	0.71
	0.71	0.71	0.71	0.71	0.89	1.08	1.26	1.44	1.63	1.81
	1.99	2.17	2.35	2.52	2.70	2.88	3.05	3.23	3.40	3.57
	3.74	3.91	4.08	4.25	4.42	4.59	4.76	4.92	5.09	5.07
	5.06	5.05	5.04	5.02	5.01	5.00	4.99	4.97	4.96	4.95
	4.94	4.93	4.91	4.90	4.89	4.88	4.86	4.85	4.84	4.83
	4.82	4.80	4.79	4.78	4.77	4.76	4.74	4.73	4.72	4.53
	4.33	4.14	3.94	3.75	3.56	3.37	3.18	2.99	2.81	2.62
	2.44	2.25	2.07	1.88	1.70	1.52	1.34	1.16	0.99	
0.74	0.73	0.73	0.73	0.73	0.73	0.73	0.73	0.72	0.72	0.72
	0.72	0.72	0.72	0.72	0.71	0.71	0.71	0.71	0.71	0.71
	0.71	0.70	0.70	0.70	0.88	1.07	1.25	1.43	1.61	1.79
	1.97	2.14	2.32	2.49	2.67	2.84	3.02	3.19	3.36	3.53
	3.70	3.87	4.04	4.20	4.37	4.53	4.70	4.86	5.02	5.01
	5.00	4.99	4.97	4.96	4.95	4.94	4.93	4.91	4.90	4.89
	4.88	4.86	4.85	4.84	4.83	4.82	4.80	4.79	4.78	4.77
	4.76	4.74	4.73	4.72	4.71	4.70	4.69	4.67	4.66	4.47
	4.28	4.09	3.89	3.71	3.52	3.33	3.14	2.96	2.77	2.59
	2.41	2.22	2.04	1.86	1.68	1.50	1.33	1.15	0.97	

0.74 0.73 0.73 0.73 0.73 0.73 0.73 0.73 0.72 0.72 0.72
0.72 0.72 0.72 0.72 0.71 0.71 0.71 0.71 0.71 0.71
0.71 0.70 0.70 0.70 0.88 1.07 1.25 1.43 1.61 1.79
1.97 2.14 2.32 2.49 2.67 2.84 3.02 3.19 3.36 3.53
3.70 3.87 4.04 4.20 4.37 4.53 4.70 4.86 5.02 5.01
5.00 4.99 4.97 4.96 4.95 4.94 4.93 4.91 4.90 4.89
4.88 4.86 4.85 4.84 4.83 4.82 4.80 4.79 4.78 4.77
4.76 4.74 4.73 4.72 4.71 4.70 4.69 4.67 4.66 4.47
4.28 4.09 3.89 3.71 3.52 3.33 3.14 2.96 2.77 2.59
2.41 2.22 2.04 1.86 1.68 1.50 1.33 1.15 0.97

Effective Rainfall (mm)

0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

Length Development Stages (days)

25.0 25.0 30.0 20.0

Basal Crop Coefficient

0.15 1.14 0.25

Rooting depth of crop (m)

0.15 0.80

Allowable depletion fraction

0.45

APPENDIX F DATA FILE INPUT FOR PUGAL SYSTEM

Table F-1 Data File Input for Pugal System

Data Input and Comments
Number of canal
15
Canal Number
1 2 3 4 5 6 7 8 9 10 11 12
13 14 15
Scheme area (ha)
892.0 126.0 184.0 1684.0 5050.0 3270.0 5584.0 12720.0 685.0 2286.0 378.0
565.0 967.0 5390.0 5123.0 20.379
Main canal full supply (m3/sec)
0.344 0.049 0.072 0.676 2.014 1.272 2.265 5.249 0.328 0.897 0.1480.224
0.395 2.236 2.018
Secondary canal full supply (m3/sec)
0.344 0.049 0.072 0.676 2.014 1.272 2.265 5.249 0.328 0.897 0.148
0.224 0.395 2.236 2.018
Stage number
1 2 3 4 5 6 7 8 9 10 11
12 13 14 15 16 17 18 19 20 21 22
23 24 25 26 27 28 29 30 31 32 33
34 35 36 37 38 39 40 41 42 43 44
45 46 47 48 49 50 51 52 53 54 55
56 57 58 59 60 61 62 63 64 65 66
67 68 69 70 71 72 73 74 75 76 77
78 79 80 81 82 83 84 85 86 87 88
89 90 91 92 93 94 95 96 97 98 99
100 101 102 103 104 105 106 107 108 109 110

111	112	113	114	115	116	117	118	119	120	121
122	123	124	125	126	127	128	129	130	131	132
133	134	135	136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151	152	153	154
155	156	157	158	159	160	161	162	163	164	165
166	167	168	169	170	171	172	173	174	175	176
177	178	179	180	181	182	183	184	185	186	187
188	189	190	191	192	193	194	195			

Number of day in this stage

1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1									

CWR (mm/day)

1.03	1.04	1.05	1.06	1.07	1.08	1.09	1.09	1.10	1.11	1.12
	1.13	1.14	1.15	1.16	1.16	1.17	1.18	1.19	1.20	1.20
	1.21	1.22	1.23	1.23	1.24	1.25	1.26	1.26	1.27	1.44

	1.62	1.80	1.98	2.16	2.34	2.52	2.70	2.89	3.08	3.26
	3.45	3.64	3.83	4.03	4.22	4.41	4.61	4.80	5.00	5.20
	5.39	5.59	5.79	5.99	6.19	6.39	6.58	6.78	6.98	7.18
	7.38	7.58	7.78	7.98	8.18	8.38	8.57	8.77	8.97	9.16
	9.36	9.55	9.75	9.94	10.13	10.32	10.51	10.70	10.89	10.88
	10.88	10.87	10.87	10.86	10.85	10.84	10.83	10.82	10.81	10.79
	10.78	10.76	10.74	10.72	10.70	10.68	10.66	10.63	10.61	10.58
	10.56	10.53	10.50	10.47	10.44	10.41	10.37	10.34	10.30	10.27
	10.23	10.19	10.15	10.11	10.07	10.03	9.99	9.94	9.90	9.85
	9.80	9.76	9.71	9.66	9.61	9.56	9.50	9.45	9.40	9.34
	9.29	9.23	9.18	9.12	9.06	9.00	8.94	8.88	8.82	8.66
	8.49	8.33	8.17	8.01	7.85	7.69	7.54	7.38	7.22	7.07
	6.92	6.77	6.62	6.47	6.32	6.17	6.03	5.88	5.74	5.60
	5.46	5.32	5.19	5.05	4.92	4.79	4.66	4.53	4.40	4.28
	4.15	4.03	3.91	3.79	3.68	3.56	3.45	3.34	3.23	3.12
	3.01	2.91	2.80	2.70	2.60	2.51	2.41	2.32	2.22	2.13
	2.04	1.96	1.87	1.79						
1.04	1.05	1.06	1.07	1.08	1.09	1.09	1.10	1.11	1.12	1.13
	1.14	1.15	1.16	1.16	1.17	1.18	1.19	1.20	1.20	1.21
	1.22	1.23	1.23	1.24	1.25	1.26	1.26	1.27	1.28	1.45
	1.63	1.81	1.99	2.17	2.35	2.53	2.72	2.90	3.09	3.28
	3.47	3.66	3.85	4.04	4.23	4.43	4.62	4.82	5.01	5.21
	5.41	5.61	5.80	6.00	6.20	6.40	6.60	6.80	7.00	7.20
	7.39	7.59	7.79	7.99	8.19	8.39	8.58	8.78	8.97	9.17
	9.36	9.56	9.75	9.94	10.13	10.32	10.51	10.70	10.88	10.88
	10.87	10.87	10.86	10.85	10.84	10.83	10.82	10.81	10.79	10.78
	10.76	10.74	10.72	10.70	10.68	10.66	10.63	10.61	10.58	10.56
	10.53	10.50	10.47	10.44	10.41	10.37	10.34	10.30	10.27	10.23
	10.19	10.15	10.11	10.07	10.03	9.99	9.94	9.90	9.85	9.80
	9.76	9.71	9.66	9.61	9.56	9.50	9.45	9.40	9.34	9.29
	9.23	9.18	9.12	9.06	9.00	8.94	8.88	8.82	8.76	8.60
	8.43	8.27	8.11	7.95	7.79	7.63	7.48	7.32	7.17	7.01

	6.86	6.71	6.56	6.41	6.26	6.12	5.97	5.83	5.69	5.55
	5.41	5.27	5.14	5.00	4.87	4.74	4.61	4.48	4.36	4.23
	4.11	3.99	3.87	3.75	3.64	3.52	3.41	3.30	3.19	3.08
	2.98	2.87	2.77	2.67	2.57	2.48	2.38	2.29	2.20	2.11
	2.02	1.93	1.85	1.76						
1.05	1.06	1.07	1.08	1.09	1.09	1.10	1.11	1.12	1.13	1.14
	1.15	1.16	1.16	1.17	1.18	1.19	1.20	1.20	1.21	1.22
	1.23	1.23	1.24	1.25	1.26	1.26	1.27	1.28	1.28	1.46
	1.64	1.81	1.99	2.18	2.36	2.54	2.73	2.91	3.10	3.29
	3.48	3.67	3.86	4.06	4.25	4.44	4.64	4.83	5.03	5.23
	5.42	5.62	5.82	6.02	6.22	6.41	6.61	6.81	7.01	7.21
	7.41	7.60	7.80	8.00	8.20	8.39	8.59	8.79	8.98	9.17
	9.37	9.56	9.75	9.94	10.13	10.32	10.51	10.69	10.88	10.87
	10.87	10.86	10.85	10.84	10.83	10.82	10.81	10.79	10.78	10.76
	10.74	10.72	10.70	10.68	10.66	10.63	10.61	10.58	10.56	10.53
	10.50	10.47	10.44	10.41	10.37	10.34	10.30	10.27	10.23	10.19
	10.15	10.11	10.07	10.03	9.99	9.94	9.90	9.85	9.80	9.76
	9.71	9.66	9.61	9.56	9.50	9.45	9.40	9.34	9.29	9.23
	9.18	9.12	9.06	9.00	8.94	8.88	8.82	8.76	8.70	8.54
	8.37	8.21	8.05	7.89	7.73	7.57	7.42	7.26	7.11	6.95
	6.80	6.65	6.50	6.35	6.21	6.06	5.92	5.78	5.63	5.49
	5.36	5.22	5.09	4.95	4.82	4.69	4.56	4.44	4.31	4.19
	4.06	3.94	3.83	3.71	3.59	3.48	3.37	3.26	3.15	3.05
	2.94	2.84	2.74	2.64	2.54	2.44	2.35	2.26	2.17	2.08
	1.99	1.91	1.82	1.74						
1.06	1.07	1.08	1.09	1.09	1.10	1.11	1.12	1.13	1.14	1.15
	1.16	1.16	1.17	1.18	1.19	1.20	1.20	1.21	1.22	1.23
	1.23	1.24	1.25	1.26	1.26	1.27	1.28	1.28	1.29	1.47
	1.64	1.82	2.00	2.19	2.37	2.55	2.74	2.93	3.11	3.30
	3.49	3.68	3.88	4.07	4.26	4.46	4.65	4.85	5.04	5.24
	5.44	5.64	5.83	6.03	6.23	6.43	6.63	6.82	7.02	7.22
	7.42	7.62	7.81	8.01	8.21	8.40	8.60	8.79	8.99	9.18

	9.37	9.56	9.75	9.94	10.13	10.32	10.51	10.69	10.87	10.87
	10.86	10.85	10.84	10.83	10.82	10.81	10.79	10.78	10.76	10.74
	10.72	10.70	10.68	10.66	10.63	10.61	10.58	10.56	10.53	10.50
	10.47	10.44	10.41	10.37	10.34	10.30	10.27	10.23	10.19	10.15
	10.11	10.07	10.03	9.99	9.94	9.90	9.85	9.80	9.76	9.71
	9.66	9.61	9.56	9.50	9.45	9.40	9.34	9.29	9.23	9.18
	9.12	9.06	9.00	8.94	8.88	8.82	8.76	8.70	8.64	8.47
	8.31	8.15	7.99	7.83	7.67	7.51	7.36	7.20	7.05	6.89
	6.74	6.59	6.44	6.30	6.15	6.01	5.86	5.72	5.58	5.44
	5.30	5.17	5.03	4.90	4.77	4.64	4.51	4.39	4.26	4.14
	4.02	3.90	3.78	3.67	3.55	3.44	3.33	3.22	3.11	3.01
	2.91	2.80	2.70	2.60	2.51	2.41	2.32	2.23	2.14	2.05
	1.97	1.88	1.80	1.72						
1.07	1.08	1.09	1.09	1.10	1.11	1.12	1.13	1.14	1.15	1.16
	1.16	1.17	1.18	1.19	1.20	1.20	1.21	1.22	1.23	1.23
	1.24	1.25	1.26	1.26	1.27	1.28	1.28	1.29	1.30	1.47
	1.65	1.83	2.01	2.20	2.38	2.56	2.75	2.94	3.13	3.32
	3.51	3.70	3.89	4.08	4.28	4.47	4.67	4.86	5.06	5.25
	5.45	5.65	5.85	6.04	6.24	6.44	6.64	6.84	7.03	7.23
	7.43	7.63	7.82	8.02	8.21	8.41	8.60	8.80	8.99	9.18
	9.37	9.57	9.75	9.94	10.13	10.32	10.50	10.69	10.87	10.86
	10.85	10.84	10.83	10.82	10.81	10.79	10.78	10.76	10.74	10.72
	10.70	10.68	10.66	10.63	10.61	10.58	10.56	10.53	10.50	10.47
	10.44	10.41	10.37	10.34	10.30	10.27	10.23	10.19	10.15	10.11
	10.07	10.03	9.99	9.94	9.90	9.85	9.80	9.76	9.71	9.66
	9.61	9.56	9.50	9.45	9.40	9.34	9.29	9.23	9.18	9.12
	9.06	9.00	8.94	8.88	8.82	8.76	8.70	8.64	8.57	8.41
	8.25	8.09	7.93	7.77	7.61	7.45	7.30	7.14	6.99	6.83
	6.68	6.53	6.39	6.24	6.09	5.95	5.81	5.67	5.53	5.39
	5.25	5.12	4.98	4.85	4.72	4.59	4.47	4.34	4.22	4.10
	3.98	3.86	3.74	3.63	3.51	3.40	3.29	3.18	3.08	2.97

	2.87	2.77	2.67	2.57	2.48	2.38	2.29	2.20	2.11	2.03
	1.94	1.86	1.78	1.70						
1.08	1.09	1.09	1.10	1.11	1.12	1.13	1.14	1.15	1.16	1.16
	1.17	1.18	1.19	1.20	1.20	1.21	1.22	1.23	1.23	1.24
	1.25	1.26	1.26	1.27	1.28	1.28	1.29	1.30	1.30	1.48
	1.66	1.84	2.02	2.20	2.39	2.57	2.76	2.95	3.14	3.33
	3.52	3.71	3.90	4.10	4.29	4.48	4.68	4.87	5.07	5.27
	5.46	5.66	5.86	6.06	6.25	6.45	6.65	6.85	7.04	7.24
	7.44	7.63	7.83	8.03	8.22	8.42	8.61	8.80	8.99	9.19
	9.38	9.57	9.75	9.94	10.13	10.31	10.50	10.68	10.86	10.85
	10.84	10.83	10.82	10.81	10.79	10.78	10.76	10.74	10.72	10.70
	10.68	10.66	10.63	10.61	10.58	10.56	10.53	10.50	10.47	10.44
	10.41	10.37	10.34	10.30	10.27	10.23	10.19	10.15	10.11	10.07
	10.03	9.99	9.94	9.90	9.85	9.80	9.76	9.71	9.66	9.61
	9.56	9.50	9.45	9.40	9.34	9.29	9.23	9.18	9.12	9.06
	9.00	8.94	8.88	8.82	8.76	8.70	8.64	8.57	8.51	8.35
	8.18	8.02	7.86	7.70	7.55	7.39	7.23	7.08	6.93	6.78
	6.62	6.48	6.33	6.18	6.04	5.89	5.75	5.61	5.47	5.34
	5.20	5.07	4.93	4.80	4.67	4.54	4.42	4.29	4.17	4.05
	3.93	3.81	3.70	3.58	3.47	3.36	3.25	3.15	3.04	2.94
	2.83	2.73	2.64	2.54	2.45	2.35	2.26	2.17	2.08	2.00
	1.91	1.83	1.75	1.67						
1.09	1.09	1.10	1.11	1.12	1.13	1.14	1.15	1.16	1.16	1.17
	1.18	1.19	1.20	1.20	1.21	1.22	1.23	1.23	1.24	1.25
	1.26	1.26	1.27	1.28	1.28	1.29	1.30	1.30	1.31	1.49
	1.67	1.85	2.03	2.21	2.40	2.58	2.77	2.96	3.15	3.34
	3.53	3.72	3.91	4.11	4.30	4.50	4.69	4.89	5.08	5.28
	5.48	5.67	5.87	6.07	6.27	6.46	6.66	6.86	7.05	7.25
	7.45	7.64	7.84	8.03	8.23	8.42	8.61	8.81	9.00	9.19
	9.38	9.57	9.75	9.94	10.12	10.31	10.49	10.67	10.85	10.84
	10.83	10.82	10.81	10.79	10.78	10.76	10.74	10.72	10.70	10.68
	10.66	10.63	10.61	10.58	10.56	10.53	10.50	10.47	10.44	10.41

	10.37	10.34	10.30	10.27	10.23	10.19	10.15	10.11	10.07	10.03
	9.99	9.94	9.90	9.85	9.80	9.76	9.71	9.66	9.61	9.56
	9.50	9.45	9.40	9.34	9.29	9.23	9.18	9.12	9.06	9.00
	8.94	8.88	8.82	8.76	8.70	8.64	8.57	8.51	8.45	8.28
	8.12	7.96	7.80	7.64	7.48	7.33	7.17	7.02	6.87	6.72
	6.57	6.42	6.27	6.12	5.98	5.84	5.70	5.56	5.42	5.28
	5.15	5.01	4.88	4.75	4.62	4.50	4.37	4.25	4.13	4.01
	3.89	3.77	3.66	3.54	3.43	3.32	3.21	3.11	3.00	2.90
	2.80	2.70	2.60	2.51	2.41	2.32	2.23	2.14	2.06	1.97
	1.89	1.81	1.73	1.65						
1.09	1.10	1.11	1.12	1.13	1.14	1.15	1.16	1.16	1.17	1.18
	1.19	1.20	1.20	1.21	1.22	1.23	1.23	1.24	1.25	1.26
	1.26	1.27	1.28	1.28	1.29	1.30	1.30	1.31	1.32	1.49
	1.67	1.86	2.04	2.22	2.41	2.59	2.78	2.97	3.16	3.35
	3.54	3.73	3.93	4.12	4.31	4.51	4.70	4.90	5.10	5.29
	5.49	5.69	5.88	6.08	6.28	6.47	6.67	6.87	7.06	7.26
	7.45	7.65	7.84	8.04	8.23	8.42	8.62	8.81	9.00	9.19
	9.38	9.56	9.75	9.94	10.12	10.30	10.48	10.66	10.84	10.83
	10.82	10.81	10.79	10.78	10.76	10.74	10.72	10.70	10.68	10.66
	10.63	10.61	10.58	10.56	10.53	10.50	10.47	10.44	10.41	10.37
	10.34	10.30	10.27	10.23	10.19	10.15	10.11	10.07	10.03	9.99
	9.94	9.90	9.85	9.80	9.76	9.71	9.66	9.61	9.56	9.50
	9.45	9.40	9.34	9.29	9.23	9.18	9.12	9.06	9.00	8.94
	8.88	8.82	8.76	8.70	8.64	8.57	8.51	8.45	8.38	8.22
	8.06	7.90	7.74	7.58	7.42	7.27	7.11	6.96	6.81	6.66
	6.51	6.36	6.21	6.07	5.92	5.78	5.64	5.50	5.36	5.23
	5.09	4.96	4.83	4.70	4.57	4.45	4.32	4.20	4.08	3.96
	3.84	3.73	3.61	3.50	3.39	3.28	3.17	3.07	2.97	2.86
	2.76	2.67	2.57	2.48	2.38	2.29	2.20	2.12	2.03	1.95
	1.86	1.78	1.70	1.63						
1.10	1.11	1.12	1.13	1.14	1.15	1.16	1.16	1.17	1.18	1.19
	1.20	1.20	1.21	1.22	1.23	1.23	1.24	1.25	1.26	1.26

	1.27	1.28	1.28	1.29	1.30	1.30	1.31	1.32	1.32	1.50
	1.68	1.86	2.05	2.23	2.42	2.60	2.79	2.98	3.17	3.36
	3.55	3.75	3.94	4.13	4.33	4.52	4.72	4.91	5.11	5.30
	5.50	5.70	5.89	6.09	6.29	6.48	6.68	6.87	7.07	7.27
	7.46	7.66	7.85	8.04	8.24	8.43	8.62	8.81	9.00	9.19
	9.38	9.56	9.75	9.93	10.11	10.30	10.48	10.65	10.83	10.82
	10.81	10.79	10.78	10.76	10.74	10.72	10.70	10.68	10.66	10.63
	10.61	10.58	10.56	10.53	10.50	10.47	10.44	10.41	10.37	10.34
	10.30	10.27	10.23	10.19	10.15	10.11	10.07	10.03	9.99	9.94
	9.90	9.85	9.80	9.76	9.71	9.66	9.61	9.56	9.50	9.45
	9.40	9.34	9.29	9.23	9.18	9.12	9.06	9.00	8.94	8.88
	8.82	8.76	8.70	8.64	8.57	8.51	8.45	8.38	8.32	8.15
	7.99	7.83	7.67	7.52	7.36	7.20	7.05	6.90	6.75	6.60
	6.45	6.30	6.15	6.01	5.87	5.72	5.58	5.45	5.31	5.17
	5.04	4.91	4.78	4.65	4.52	4.40	4.27	4.15	4.03	3.91
	3.80	3.68	3.57	3.46	3.35	3.24	3.14	3.03	2.93	2.83
	2.73	2.63	2.54	2.44	2.35	2.26	2.17	2.09	2.00	1.92
	1.84	1.76	1.68	1.61						
1.11	1.12	1.13	1.14	1.15	1.16	1.16	1.17	1.18	1.19	1.20
	1.20	1.21	1.22	1.23	1.23	1.24	1.25	1.26	1.26	1.27
	1.28	1.28	1.29	1.30	1.30	1.31	1.32	1.32	1.33	1.51
	1.69	1.87	2.06	2.24	2.43	2.61	2.80	2.99	3.18	3.37
	3.56	3.76	3.95	4.14	4.34	4.53	4.73	4.92	5.12	5.31
	5.51	5.71	5.90	6.10	6.30	6.49	6.69	6.88	7.08	7.27
	7.47	7.66	7.85	8.05	8.24	8.43	8.62	8.81	9.00	9.19
	9.37	9.56	9.74	9.93	10.11	10.29	10.47	10.64	10.82	10.81
	10.79	10.78	10.76	10.74	10.72	10.70	10.68	10.66	10.63	10.61
	10.58	10.56	10.53	10.50	10.47	10.44	10.41	10.37	10.34	10.30
	10.27	10.23	10.19	10.15	10.11	10.07	10.03	9.99	9.94	9.90
	9.85	9.80	9.76	9.71	9.66	9.61	9.56	9.50	9.45	9.40
	9.34	9.29	9.23	9.18	9.12	9.06	9.00	8.94	8.88	8.82
	8.76	8.70	8.64	8.57	8.51	8.45	8.38	8.32	8.25	8.09

	7.93	7.77	7.61	7.45	7.30	7.14	6.99	6.84	6.68	6.53
	6.39	6.24	6.09	5.95	5.81	5.67	5.53	5.39	5.26	5.12
	4.99	4.86	4.73	4.60	4.47	4.35	4.23	4.11	3.99	3.87
	3.75	3.64	3.53	3.42	3.31	3.20	3.10	2.99	2.89	2.79
	2.70	2.60	2.51	2.41	2.32	2.23	2.15	2.06	1.98	1.89
	1.81	1.74	1.66	1.58						
1.12	1.13	1.14	1.15	1.16	1.16	1.17	1.18	1.19	1.20	1.20
	1.21	1.22	1.23	1.23	1.24	1.25	1.26	1.26	1.27	1.28
	1.28	1.29	1.30	1.30	1.31	1.32	1.32	1.33	1.33	1.51
	1.70	1.88	2.06	2.25	2.44	2.62	2.81	3.00	3.19	3.38
	3.57	3.77	3.96	4.15	4.35	4.54	4.74	4.93	5.13	5.32
	5.52	5.72	5.91	6.11	6.30	6.50	6.70	6.89	7.08	7.28
	7.47	7.67	7.86	8.05	8.24	8.43	8.62	8.81	9.00	9.18
	9.37	9.55	9.74	9.92	10.10	10.28	10.46	10.63	10.81	10.79
	10.78	10.76	10.74	10.72	10.70	10.68	10.66	10.63	10.61	10.58
	10.56	10.53	10.50	10.47	10.44	10.41	10.37	10.34	10.30	10.27
	10.23	10.19	10.15	10.11	10.07	10.03	9.99	9.94	9.90	9.85
	9.80	9.76	9.71	9.66	9.61	9.56	9.50	9.45	9.40	9.34
	9.29	9.23	9.18	9.12	9.06	9.00	8.94	8.88	8.82	8.76
	8.70	8.64	8.57	8.51	8.45	8.38	8.32	8.25	8.19	8.02
	7.86	7.70	7.55	7.39	7.23	7.08	6.92	6.77	6.62	6.47
	6.33	6.18	6.04	5.89	5.75	5.61	5.47	5.34	5.20	5.07
	4.94	4.81	4.68	4.55	4.42	4.30	4.18	4.06	3.94	3.82
	3.71	3.60	3.49	3.38	3.27	3.16	3.06	2.96	2.86	2.76
	2.66	2.57	2.47	2.38	2.29	2.20	2.12	2.03	1.95	1.87
	1.79	1.71	1.64	1.56						
1.13	1.14	1.15	1.16	1.16	1.17	1.18	1.19	1.20	1.20	1.21
	1.22	1.23	1.23	1.24	1.25	1.26	1.26	1.27	1.28	1.28
	1.29	1.30	1.30	1.31	1.32	1.32	1.33	1.33	1.34	1.52
	1.70	1.89	2.07	2.26	2.44	2.63	2.82	3.01	3.20	3.39
	3.58	3.78	3.97	4.16	4.36	4.55	4.75	4.94	5.14	5.33
	5.53	5.73	5.92	6.12	6.31	6.51	6.70	6.90	7.09	7.28

	7.48	7.67	7.86	8.05	8.24	8.43	8.62	8.81	9.00	9.18
	9.37	9.55	9.73	9.91	10.09	10.27	10.44	10.62	10.79	10.78
	10.76	10.74	10.72	10.70	10.68	10.66	10.63	10.61	10.58	10.56
	10.53	10.50	10.47	10.44	10.41	10.37	10.34	10.30	10.27	10.23
	10.19	10.15	10.11	10.07	10.03	9.99	9.94	9.90	9.85	9.80
	9.76	9.71	9.66	9.61	9.56	9.50	9.45	9.40	9.34	9.29
	9.23	9.18	9.12	9.06	9.00	8.94	8.88	8.82	8.76	8.70
	8.64	8.57	8.51	8.45	8.38	8.32	8.25	8.19	8.12	7.96
	7.80	7.64	7.48	7.32	7.17	7.01	6.86	6.71	6.56	6.41
	6.27	6.12	5.98	5.83	5.69	5.55	5.42	5.28	5.15	5.01
	4.88	4.75	4.63	4.50	4.38	4.25	4.13	4.01	3.90	3.78
	3.67	3.55	3.44	3.34	3.23	3.12	3.02	2.92	2.82	2.72
	2.63	2.53	2.44	2.35	2.26	2.17	2.09	2.01	1.92	1.84
	1.77	1.69	1.61	1.54						
1.14	1.15	1.16	1.16	1.17	1.18	1.19	1.20	1.20	1.21	1.22
	1.23	1.23	1.24	1.25	1.26	1.26	1.27	1.28	1.28	1.29
	1.30	1.30	1.31	1.32	1.32	1.33	1.33	1.34	1.34	1.53
	1.71	1.89	2.08	2.27	2.45	2.64	2.83	3.02	3.21	3.40
	3.59	3.79	3.98	4.17	4.37	4.56	4.76	4.95	5.15	5.34
	5.54	5.73	5.93	6.12	6.32	6.51	6.71	6.90	7.10	7.29
	7.48	7.67	7.86	8.05	8.24	8.43	8.62	8.81	8.99	9.18
	9.36	9.54	9.72	9.90	10.08	10.26	10.43	10.60	10.78	10.76
	10.74	10.72	10.70	10.68	10.66	10.63	10.61	10.58	10.56	10.53
	10.50	10.47	10.44	10.41	10.37	10.34	10.30	10.27	10.23	10.19
	10.15	10.11	10.07	10.03	9.99	9.94	9.90	9.85	9.80	9.76
	9.71	9.66	9.61	9.56	9.50	9.45	9.40	9.34	9.29	9.23
	9.18	9.12	9.06	9.00	8.94	8.88	8.82	8.76	8.70	8.64
	8.57	8.51	8.45	8.38	8.32	8.25	8.19	8.12	8.05	7.89
	7.73	7.57	7.42	7.26	7.10	6.95	6.80	6.65	6.50	6.35
	6.21	6.06	5.92	5.78	5.64	5.50	5.36	5.23	5.09	4.96
	4.83	4.70	4.57	4.45	4.33	4.20	4.08	3.97	3.85	3.74
	3.62	3.51	3.40	3.29	3.19	3.09	2.98	2.88	2.78	2.69

	2.59	2.50	2.41	2.32	2.23	2.15	2.06	1.98	1.90	1.82
	1.74	1.67	1.59	1.52						
1.15	1.16	1.16	1.17	1.18	1.19	1.20	1.20	1.21	1.22	1.23
	1.23	1.24	1.25	1.26	1.26	1.27	1.28	1.28	1.29	1.30
	1.30	1.31	1.32	1.32	1.33	1.33	1.34	1.34	1.35	1.53
	1.72	1.90	2.09	2.27	2.46	2.65	2.84	3.03	3.22	3.41
	3.60	3.80	3.99	4.18	4.38	4.57	4.77	4.96	5.16	5.35
	5.55	5.74	5.94	6.13	6.33	6.52	6.71	6.91	7.10	7.29
	7.48	7.68	7.87	8.06	8.24	8.43	8.62	8.80	8.99	9.17
	9.35	9.53	9.71	9.89	10.07	10.24	10.42	10.59	10.76	10.74
	10.72	10.70	10.68	10.66	10.63	10.61	10.58	10.56	10.53	10.50
	10.47	10.44	10.41	10.37	10.34	10.30	10.27	10.23	10.19	10.15
	10.11	10.07	10.03	9.99	9.94	9.90	9.85	9.80	9.76	9.71
	9.66	9.61	9.56	9.50	9.45	9.40	9.34	9.29	9.23	9.18
	9.12	9.06	9.00	8.94	8.88	8.82	8.76	8.70	8.64	8.57
	8.51	8.45	8.38	8.32	8.25	8.19	8.12	8.05	7.98	7.82
	7.66	7.51	7.35	7.19	7.04	6.89	6.74	6.59	6.44	6.29
	6.14	6.00	5.86	5.72	5.58	5.44	5.30	5.17	5.04	4.91
	4.78	4.65	4.52	4.40	4.28	4.16	4.04	3.92	3.80	3.69
	3.58	3.47	3.36	3.25	3.15	3.05	2.95	2.85	2.75	2.65
	2.56	2.47	2.38	2.29	2.20	2.12	2.03	1.95	1.87	1.79
	1.72	1.64	1.57	1.50						
1.16	1.16	1.17	1.18	1.19	1.20	1.20	1.21	1.22	1.23	1.23
	1.24	1.25	1.26	1.26	1.27	1.28	1.28	1.29	1.30	1.30
	1.31	1.32	1.32	1.33	1.33	1.34	1.34	1.35	1.36	1.54
	1.72	1.91	2.09	2.28	2.47	2.66	2.85	3.04	3.23	3.42
	3.61	3.81	4.00	4.19	4.39	4.58	4.78	4.97	5.17	5.36
	5.55	5.75	5.94	6.14	6.33	6.53	6.72	6.91	7.10	7.30
	7.49	7.68	7.87	8.05	8.24	8.43	8.61	8.80	8.98	9.17
	9.35	9.53	9.70	9.88	10.06	10.23	10.40	10.57	10.74	10.72
	10.70	10.68	10.66	10.63	10.61	10.58	10.56	10.53	10.50	10.47
	10.44	10.41	10.37	10.34	10.30	10.27	10.23	10.19	10.15	10.11

	10.07	10.03	9.99	9.94	9.90	9.85	9.80	9.76	9.71	9.66
	9.61	9.56	9.50	9.45	9.40	9.34	9.29	9.23	9.18	9.12
	9.06	9.00	8.94	8.88	8.82	8.76	8.70	8.64	8.57	8.51
	8.45	8.38	8.32	8.25	8.19	8.12	8.05	7.98	7.92	7.76
	7.60	7.44	7.28	7.13	6.98	6.82	6.67	6.52	6.37	6.23
	6.08	5.94	5.80	5.66	5.52	5.38	5.25	5.11	4.98	4.85
	4.72	4.60	4.47	4.35	4.23	4.11	3.99	3.87	3.76	3.65
	3.54	3.43	3.32	3.21	3.11	3.01	2.91	2.81	2.71	2.62
	2.53	2.43	2.35	2.26	2.17	2.09	2.01	1.93	1.85	1.77
	1.69	1.62	1.55	1.48						

Effective rainfall (mm/day)

0.33	0.25	0.17	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.17	0.34	0.50	0.65	0.79
	0.92	1.05	1.17	1.28	1.39	1.48	1.57	1.65	1.71	1.77
	1.83	1.87	1.90	1.93	1.94	1.94	1.94	1.92	1.90	1.86
	1.82	1.76	1.70	1.62	1.53	1.43	1.32	1.19	1.06	0.91
	0.76	0.59	0.40	0.21	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.13	0.19
	0.25	0.31	0.36	0.41	0.46	0.50	0.54	0.58	0.62	0.65
	0.68	0.70	0.72	0.74						

0.25	0.17	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.17	0.34	0.50	0.65	0.79	0.92
	1.05	1.17	1.28	1.39	1.48	1.57	1.65	1.71	1.77	1.83
	1.87	1.90	1.93	1.94	1.94	1.94	1.92	1.90	1.86	1.82
	1.76	1.70	1.62	1.53	1.43	1.32	1.19	1.06	0.91	0.76
	0.59	0.40	0.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.13	0.19	0.25
	0.31	0.36	0.41	0.46	0.50	0.54	0.58	0.62	0.65	0.68
	0.70	0.72	0.74	0.76						
0.17	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.17	0.34	0.50	0.65	0.79	0.92	1.05
	1.17	1.28	1.39	1.48	1.57	1.65	1.71	1.77	1.83	1.87
	1.90	1.93	1.94	1.94	1.94	1.92	1.90	1.86	1.82	1.76
	1.70	1.62	1.53	1.43	1.32	1.19	1.06	0.91	0.76	0.59
	0.40	0.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.07	0.13	0.19	0.25	0.31
	0.36	0.41	0.46	0.50	0.54	0.58	0.62	0.65	0.68	0.70
	0.72	0.74	0.76	0.77						
0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.17	0.34	0.50	0.65	0.79	0.92	1.05	1.17
	1.28	1.39	1.48	1.57	1.65	1.71	1.77	1.83	1.87	1.90
	1.93	1.94	1.94	1.94	1.92	1.90	1.86	1.82	1.76	1.70
	1.62	1.53	1.43	1.32	1.19	1.06	0.91	0.76	0.59	0.40
	0.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.07	0.13	0.19	0.25	0.31	0.36
	0.41	0.46	0.50	0.54	0.58	0.62	0.65	0.68	0.70	0.72
	0.74	0.76	0.77	0.77						
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.17	0.34	0.50	0.65	0.79	0.92	1.05	1.17	1.28
	1.39	1.48	1.57	1.65	1.71	1.77	1.83	1.87	1.90	1.93
	1.94	1.94	1.94	1.92	1.90	1.86	1.82	1.76	1.70	1.62
	1.53	1.43	1.32	1.19	1.06	0.91	0.76	0.59	0.40	0.21
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.07	0.13	0.19	0.25	0.31	0.36	0.41
	0.46	0.50	0.54	0.58	0.62	0.65	0.68	0.70	0.72	0.74
	0.76	0.77	0.77	0.78						
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.17	0.34	0.50	0.65	0.79	0.92	1.05	1.17	1.28	1.39
	1.48	1.57	1.65	1.71	1.77	1.83	1.87	1.90	1.93	1.94
	1.94	1.94	1.92	1.90	1.86	1.82	1.76	1.70	1.62	1.53
	1.43	1.32	1.19	1.06	0.91	0.76	0.59	0.40	0.21	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.07	0.13	0.19	0.25	0.31	0.36	0.41	0.46
	0.50	0.54	0.58	0.62	0.65	0.68	0.70	0.72	0.74	0.76
	0.77	0.77	0.78	0.78						
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.17
	0.34	0.50	0.65	0.79	0.92	1.05	1.17	1.28	1.39	1.48
	1.57	1.65	1.71	1.77	1.83	1.87	1.90	1.93	1.94	1.94
	1.94	1.92	1.90	1.86	1.82	1.76	1.70	1.62	1.53	1.43
	1.32	1.19	1.06	0.91	0.76	0.59	0.40	0.21	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.07	0.13	0.19	0.25	0.31	0.36	0.41	0.46	0.50
	0.54	0.58	0.62	0.65	0.68	0.70	0.72	0.74	0.76	0.77
	0.77	0.78	0.78	0.77						
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.17	0.34
	0.50	0.65	0.79	0.92	1.05	1.17	1.28	1.39	1.48	1.57
	1.65	1.71	1.77	1.83	1.87	1.90	1.93	1.94	1.94	1.94
	1.92	1.90	1.86	1.82	1.76	1.70	1.62	1.53	1.43	1.32

	1.19	1.06	0.91	0.76	0.59	0.40	0.21	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.07	0.13	0.19	0.25	0.31	0.36	0.41	0.46	0.50	0.54
	0.58	0.62	0.65	0.68	0.70	0.72	0.74	0.76	0.77	0.77
	0.78	0.78	0.77	0.77						
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.17	0.34	0.50
	0.65	0.79	0.92	1.05	1.17	1.28	1.39	1.48	1.57	1.65
	1.71	1.77	1.83	1.87	1.90	1.93	1.94	1.94	1.94	1.92
	1.90	1.86	1.82	1.76	1.70	1.62	1.53	1.43	1.32	1.19
	1.06	0.91	0.76	0.59	0.40	0.21	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07
	0.13	0.19	0.25	0.31	0.36	0.41	0.46	0.50	0.54	0.58
	0.62	0.65	0.68	0.70	0.72	0.74	0.76	0.77	0.77	0.78
	0.78	0.77	0.77	0.76						

0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.17	0.34	0.50	0.65
	0.79	0.92	1.05	1.17	1.28	1.39	1.48	1.57	1.65	1.71
	1.77	1.83	1.87	1.90	1.93	1.94	1.94	1.94	1.92	1.90
	1.86	1.82	1.76	1.70	1.62	1.53	1.43	1.32	1.19	1.06
	0.91	0.76	0.59	0.40	0.21	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.13
	0.19	0.25	0.31	0.36	0.41	0.46	0.50	0.54	0.58	0.62
	0.65	0.68	0.70	0.72	0.74	0.76	0.77	0.77	0.78	0.78
	0.77	0.77	0.76	0.74						
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.17	0.34	0.50	0.65	0.79
	0.92	1.05	1.17	1.28	1.39	1.48	1.57	1.65	1.71	1.77
	1.83	1.87	1.90	1.93	1.94	1.94	1.94	1.92	1.90	1.86
	1.82	1.76	1.70	1.62	1.53	1.43	1.32	1.19	1.06	0.91
	0.76	0.59	0.40	0.21	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.13	0.19
	0.25	0.31	0.36	0.41	0.46	0.50	0.54	0.58	0.62	0.65
	0.68	0.70	0.72	0.74	0.76	0.77	0.77	0.78	0.78	0.77
	0.77	0.76	0.74	0.72						
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.17	0.34	0.50	0.65	0.79	0.92
	1.05	1.17	1.28	1.39	1.48	1.57	1.65	1.71	1.77	1.83
	1.87	1.90	1.93	1.94	1.94	1.94	1.92	1.90	1.86	1.82
	1.76	1.70	1.62	1.53	1.43	1.32	1.19	1.06	0.91	0.76
	0.59	0.40	0.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.13	0.19	0.25
	0.31	0.36	0.41	0.46	0.50	0.54	0.58	0.62	0.65	0.68
	0.70	0.72	0.74	0.76	0.77	0.77	0.78	0.78	0.77	0.77
	0.76	0.74	0.72	0.70						
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

	0.00	0.00	0.00	0.17	0.34	0.50	0.65	0.79	0.92	1.05
	1.17	1.28	1.39	1.48	1.57	1.65	1.71	1.77	1.83	1.87
	1.90	1.93	1.94	1.94	1.94	1.92	1.90	1.86	1.82	1.76
	1.70	1.62	1.53	1.43	1.32	1.19	1.06	0.91	0.76	0.59
	0.40	0.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.07	0.13	0.19	0.25	0.31
	0.36	0.41	0.46	0.50	0.54	0.58	0.62	0.65	0.68	0.70
	0.72	0.74	0.76	0.77	0.77	0.78	0.78	0.77	0.77	0.76
	0.74	0.72	0.70	0.67						
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.17	0.34	0.50	0.65	0.79	0.92	1.05	1.17
	1.28	1.39	1.48	1.57	1.65	1.71	1.77	1.83	1.87	1.90
	1.93	1.94	1.94	1.94	1.92	1.90	1.86	1.82	1.76	1.70
	1.62	1.53	1.43	1.32	1.19	1.06	0.91	0.76	0.59	0.40
	0.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

	0.00	0.00	0.00	0.00	0.07	0.13	0.19	0.25	0.31	0.36
	0.41	0.46	0.50	0.54	0.58	0.62	0.65	0.68	0.70	0.72
	0.74	0.76	0.77	0.77	0.78	0.78	0.77	0.77	0.76	0.74
	0.72	0.70	0.67	0.64						
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.17	0.34	0.50	0.65	0.79	0.92	1.05	1.17	1.28
	1.39	1.48	1.57	1.65	1.71	1.77	1.83	1.87	1.90	1.93
	1.94	1.94	1.94	1.92	1.90	1.86	1.82	1.76	1.70	1.62
	1.53	1.43	1.32	1.19	1.06	0.91	0.76	0.59	0.40	0.21
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.07	0.13	0.19	0.25	0.31	0.36	0.41
	0.46	0.50	0.54	0.58	0.62	0.65	0.68	0.70	0.72	0.74
	0.76	0.77	0.77	0.78	0.78	0.77	0.77	0.76	0.74	0.72
	0.70	0.67	0.64	0.60						

Loss in gate opening by canal groups(mm)

1.160	0.401	0.219	3.778	8.429	4.850	6.621	14.737	1.373	3.597	1.714
	2.025	1.282	11.739	6.546						

Field capacity (m^3/m^3)

0.134	0.134	0.134	0.134	0.134	0.134	0.134	0.134	0.134	0.134	0.134
	0.134	0.134	0.134	0.134						

Permanent wilting point (m^3/m^3)

0.06 0.06 0.06 0.06 0.06 0.06 0.06 0.06 0.06 0.06 0.06
0.06 0.06 0.06 0.06

Length Development Stages (days)

30.0 50.0 60.0 55.0

Basal Crop Coefficient

0.15 1.13 0.4

Maximum and minimum rooting depths (m)

0.3 1.4

Allowable depletion fraction

0.65

APPENDIX G SOURCE CODE FOR WATER SCHEDULING PROBLEM

This appendix contains the source codes for water scheduling problems used to solve the test systems and the Pugal system. There are source codes for scheduling in secondary canal level (chapter 6), source code for scheduling in both secondary and tertiary canals (chapter 7), and source code for Pugal system (chapter 8). The main part of the source codes are very similar. Source codes are also set up differently for the warabandi and zero-1 approaches. The contents of the file used in the GA program for scheduling are given in tables as follows:

Table G-1 presents the file used in the GA program for scheduling at secondary canal level in chapter 6. Staggering days in each canal were pre-specified in this test system. Fixed losses for gate opening and shutting are not different in each canal.

Table G-1 Contents of the files used in the GA model (for chapter 6)

File name	**Description**
GA44.c	Name of source code file
PrjNsch3.lst	Name of input project file for non stress case
ProjDf3X.lst	Name of input project file for stress condition
WatNsch3.dat	Name of input data file for non stress case
WatDf3X.dat	Name of input data file for stress case
ga44.out	Output file for overall detail

Table G-2 presents the files used in the GA program for scheduling at both secondary canal and tertiary canal level in chapter 7 with zero-1 approach. Table G-3 contains the files used for warabandi approach.

Table G-2 Contents of the files used in the GA model for zero-1 approach (chapter 7)

File name	Description
GA46A.c	Name of source code file
Proj46A.lst	Name of input project file for zero-1 criteria
Input46A.dat	Name of input data file
ga46A.out	Output file for overall detail

Table G-3 Contents of the files used in the GA model for warabandi approach (chapter 7)

File name	Description
GA46W.c	Name of source code file
Proj46W.lst	Name of input project file for zero-1 criteria
Input46W.dat	Name of input data file
Ga46W.out	Output file for overall detail

Table G-4 presents the file used in the GA program for Pugal system in chapter 8 by zero-1 approach. Table G-5 contains the files used for warabandi approach. In chapter 8, the Pugal system is rotated only at secondary level.

Table G-4 Contents of the files used in Pugal system for zero-1 approach (chapter 8)

File name	Description
GA47A.c	Name of source code file
Proj47A.lst	Name of input project file for zero-1 criteria
Input47A.dat	Name of input data file
ga47A.out	Output file for overall detail

Table G-5 Contents of the files used in Pugal system for warabandi approach (chapter 8)

File name	**Description**
GA47W.c	Name of source code file
Proj47W.lst	Name of input project file for zero-1 criteria
Input47W.dat	Name of input data file
Ga47W.out	Output file for overall detail

```
/*GA44.c          Program to formulate GAs for water scheduling problem
                  Water Scheduling, Rotation in Secondary Only (chapter 6)
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <process.h>
#include <time.h>

#define Tperiod       1.0     // input Time Period here (0.5 or 1.0 time period)
#define POPSIZE       100     // input population size
#define NSCHEME       3       // input number of scheme
#define Calendar      100     // input total growth period
#define STAGE         100     // input total stage day (for
#define CYCLE         103     // input total time step = Calendar/Tperiod)+ total max. staggering
                              // (= 206 for 0.5 time period)
#define LENGTH        CYCLE*NSCHEME     // chromosome length
#define INTERVAL      300     // input interval of generation to recheck of improvement fitness
#define DELTA         0.0001  // input minimum improvement of fitness after INTERVAL
generations
#define IrriEff       0.7     // input irrigation efficiency
#define fixloss       1.5     // input fix loss
#define  layerD       0.005   // input a root zone soil layer for water balance in meter
#define  tDepth       1.0     // input total depth of soil layer in meter
#define MaxLyr        200     // input maximum soil layer in consider

int     RandInt (int, int);
float   RandReal (float,float);
void    Read_inputfiles();
void    SwapIntValues(int *, int *);
void    SwapRealValues(float *, float *);
void    nrerror(char *messg) ;
void    initial();
void    objective(int);
void    SelectMate();
void    UniSelectMate();
void    mutatepop();
void    Tournament();

int     i, tt, index, mated[POPSIZE], t_lag;
float   mated_pop[POPSIZE][LENGTH], initial_pop[POPSIZE][LENGTH],Pmutate;
float   ftnss[POPSIZE+1], DELTA_ftnss, OLD_minftnss, bestfit;
int     IFLAG[NSCHEME][CYCLE],  pFlag[NSCHEME][CYCLE];
int     MaxGEN, R1, R2;
float   PXOVER, NofMu;
float   countday;
int     root_period, stage_day[STAGE], stagenum[STAGE];
float   D_factr;
float   MinRD, MaxRD, Lini, Ldev, Lmid, Llate, Kcb_ini, Kcb_mid, Kcb_end, Kcb[CYCLE];
float   FC[NSCHEME], PWP[NSCHEME];
float   Bflowrate[NSCHEME], input_ETc[NSCHEME][STAGE], main_capa;
float   scheme_area[NSCHEME], tot_area;  //new
float   psumOVR, psumLWP, psumCapa;
float   sumOversupl[POPSIZE], sumCapaConstr[POPSIZE], sumLWP[POPSIZE];
float   LWP[NSCHEME][CYCLE], pLWP[NSCHEME][CYCLE];
float   Ovr_supl[NSCHEME][CYCLE], pOvr_supl[NSCHEME][CYCLE];
float   ETc[NSCHEME][CYCLE];
```

```
float     Eff_rain[STAGE];
float     sup_wrbd[NSCHEME], psup_wrbd[NSCHEME][CYCLE];
float     supply[NSCHEME][CYCLE], psupply[NSCHEME][CYCLE];
float     demand[NSCHEME][CYCLE], pdemand[NSCHEME][CYCLE];
float     capaConstr[POPSIZE][CYCLE],    pcapaConstr[CYCLE];
int       LagC[NSCHEME]; //new

FILE *f1,*f2,*f3 ;

//MAIN PROGRAM
void main (void) {
   int gen;
   clock_t start, finish;
   float  duration, TotalTime ;

   TotalTime = 0.0f;                                    //ga44trlg.out for Tperiod = 1 nonstress
   if(!(f1 = fopen("ga44trlg.out","w"))) {              //ga44Df3s.out for Tperiod = 1 stress
                  printf("can't open file output file\n");//ga44Df2s.out for Tperiod = 0.5 stress
                  exit(1);
          }
   Read_inputfiles();
          srand(2);
 start = clock();

   MaxGEN       = 2500;
          R1              = 1;
          R2              = 10000;
          PXOVER          = 0.85f;
          NofMu  = 0.05f;

/* R1   = 1;
do{
   R2   = 5000;
do{ */
/*        NofMu  = 0.01f;
do{
   PXOVER       = 0.5f;
do{ */

   initial();
   index = 0;
   OLD_minftnss = 1000.0f;
          DELTA_ftnss             = 1000.0f;
          bestfit                 = 1000000.0f;
          ftnss[POPSIZE]  = 10000000000.0f;
 for(gen = 0; gen < MaxGEN; gen++) {
          objective(gen); if(index == 999) break;
          Tournament();
//        SelectMate();
          UniSelectMate();
          mutatepop();
   if (!((gen+1)%INTERVAL)){
     DELTA_ftnss = (float)fabs((double)((OLD_minftnss - ftnss[POPSIZE])/ftnss[POPSIZE]));
     OLD_minftnss = ftnss[POPSIZE];
          }
 } //End of Generation Loop
/*PXOVER = (float)(PXOVER + 0.05f);
}while (PXOVER <= 1.01f);
```

```
 NofMu = (float)(NofMu + 0.01);
}while (NofMu  <= 0.11); */
/*R2 = R2 + 2500;
}while (R2 <= 12501);
 R1 = R1 + 2;
}while (R1 <= 6); */
 finish = clock();
 duration = (float)(finish - start) / CLOCKS_PER_SEC;
 TotalTime = (float)(TotalTime + duration);
(void) fclose (f1);
 printf("End of run  ENJOY!!!! ");
}  //MAIN PROGRAM END
/*******************/
void Read_inputfiles() {
int  ii, jj, num_schemes, scheme_node[NSCHEME];
char  title[80],title2[80],title3[80],title4[80],title5[80],title6[80],title8[80];
char  title9[80], title10[80], title11[80], title12[80], title13[80], schedulefile[20];

if(!(f2 = fopen("prjNsch3.lst","r"))) {   //prjNsch3.lst for watNsch3.dat sandy nonstress
    nrerror(" Can not open project file") ; //projDf3X.lst for watDef3X.dat sandy stress 1 and 0.5 day
t-period
}
// now read the data file names
        fgets(schedulefile, 20, f2) ;
        for(ii = 0; ii < 20; ii++) {
                if(schedulefile[ii] == '\n') schedulefile[ii] = '\0' ;
        }
        (void) fclose(f2) ;
if(!(f2 = fopen(schedulefile,"r"))) {
    nrerror(" Unable to open schedulefile") ;
}
fgets(title, 80, f2);               // Water scheduling title
fscanf(f2,"%d",&num_schemes);  // scan number of schemes here
for(jj=0;jj<num_schemes;jj++) fscanf(f2,"%d ", &scheme_node[jj]);
fgets(title2, 80, f2);        // scheme area (ha) title
//        fputs(title2,f1);
for(jj=0;jj<num_schemes;jj++) fscanf(f2,"%f ", &scheme_area[jj]);
        tot_area = 0.0f;
        for(jj=0;jj<num_schemes;jj++){
                tot_area = tot_area + scheme_area[jj];  // new
        }
fgets(title13, 80, f2);        // staggering days between schemes
for(jj=0;jj<num_schemes;jj++) fscanf(f2,"%d ", &LagC[jj]);
fgets(title3, 80, f2);        // main canal supply title
        fscanf(f2,"%f ", &main_capa) ;  // scan main canal supply
fgets(title4, 80, f2);        // branch canal title
// scan branch canal supply
        for(jj = 0; jj < num_schemes; jj++) fscanf(f2,"%f ", &Bflowrate[jj]) ;
        fscanf(f2,"\n");
fgets(title5, 80, f2);        // CWR title
        for(tt=0; tt<STAGE; tt++) fscanf(f2,"%d ", &stagenum[tt]);
        for(tt=0; tt<STAGE; tt++) fscanf(f2,"%d ", &stage_day[tt]);
// scan input data of ETc (mm/period)
        for(jj = 0; jj < num_schemes; jj++)  {
                for(tt = 0; tt <STAGE; tt++){
                   fscanf(f2,"%f", &input_ETc[jj][tt]) ;
                }
        }
```

```
        fscanf(f2,"\n");
fgets(title6, 80, f2);        // ER title
        for(tt = 0; tt <STAGE; tt++){
           fscanf(f2,"%f", &Eff_rain[tt]) ;
        }
fscanf(f2,"\n");
fgets(title8, 80, f2);        // field capacity title
        // scan FC
        for(jj = 0; jj < num_schemes; jj++) {
                fscanf(f2,"%f ", &FC[jj]) ;
        }
fgets(title9, 80, f2);        // permanent wilting point title
        for(jj = 0; jj < num_schemes; jj++) {
                fscanf(f2,"%f ", &PWP[jj]) ;
        }
fscanf(f2,"\n");
fgets(title10,80,f2);   // Length stage title
        fscanf(f2,"%f %f %f %f",&Lini, &Ldev, &Lmid, &Llate);
        fscanf(f2,"\n");
        fgets(title11,80,f2);   // Kcb title
        fscanf(f2,"%f %f %f",&Kcb_ini, &Kcb_mid, &Kcb_end);
        fscanf(f2,"\n");
fgets(title12, 80, f2); // rooting depth title
        fscanf(f2,"%f %f", &MinRD, &MaxRD);
fscanf(f2,"\n");
        fscanf(f2,"%f",&D_factr);
(void)fclose(f2);
}
/*********/
float RandReal (float lower,float upper) {
        float val;
        float nb = ( upper - lower );
        val =  (float)(rand()%1000/1000.0*nb + lower);
        return ((float)val);
}
/*********/
int RandInt (int lower,int upper) {
        int val;
        int nb = (upper - lower );
        val =  (rand()%nb + lower);
        return val;
}
/*********/
void initial() {
  int i,j;
  float nd = RAND_MAX;

  for(i=0;i<POPSIZE;i++) {
    for(j=0;j<LENGTH;j++)  {
         initial_pop[i][j] = (float)(rand()&01);
    }
  }
}
/**********/
void  objective(int gen) {
        int     i, k, jj, ii, cycle[POPSIZE][NSCHEME], t_lag_adj;
        float   sumftnss, minftnss, avgftnss;
        float   sumcycle, odd2[POPSIZE][STAGE];
```

```
        float   branchGO[CYCLE][NSCHEME], sumBflow[CYCLE], odds[POPSIZE];
        float   GlobalSMC[POPSIZE][NSCHEME],pGlobalSMC[NSCHEME];
        float   sumX[POPSIZE][NSCHEME], psumX[NSCHEME];
        float   sumsmc_rz[POPSIZE][NSCHEME], psumsmc_rz[NSCHEME];
        float   smc_rz[NSCHEME][CYCLE], psmc_rz[NSCHEME][CYCLE];
        float   sumsupply[POPSIZE], sumGlobal[POPSIZE], sumO[POPSIZE][NSCHEME];
        float   psumsupply, psumO[NSCHEME];
        float   psumEquity, sumEquity[POPSIZE], Equity[NSCHEME][CYCLE];
        float   RD[NSCHEME][CYCLE]
        float   pSMCline[NSCHEME][CYCLE], SMCline[NSCHEME][CYCLE];
        float   P[CYCLE], Ks[NSCHEME][CYCLE], sumCROP_DIE[POPSIZE];
        int     NLR_RZ[NSCHEME][CYCLE], nlr_RZ_MAX, nsl, countn; //new
        float   FC_layer, RAM_layer, WP_layer, RAM_RZ[NSCHEME][CYCLE]; //new
        float   FC_RZ[NSCHEME][CYCLE], PWP_RZ[NSCHEME][CYCLE],
                WP_RZ[NSCHEME][CYCLE];//new
        float   smc_layer[200][NSCHEME][CYCLE], tsum[NSCHEME][CYCLE],
                ptsum[NSCHEME][CYCLE], infilt ; //new
        float   irrig[NSCHEME][CYCLE], pstress[NSCHEME],
                cum_stress[POPSIZE][NSCHEME];
        float   pRD[NSCHEME][CYCLE];
        float   psuminfil[NSCHEME], suminfil[POPSIZE][NSCHEME];
        float   pinfiltopup[NSCHEME][CYCLE], infiltopup[NSCHEME][CYCLE];
        float   ER[CYCLE];
        float   Ea[NSCHEME][CYCLE], pEa[NSCHEME][CYCLE],
                sumEa[POPSIZE][NSCHEME],psumEa[NSCHEME];
        float   sumIrr_day[POPSIZE][NSCHEME], psumIrr_day[NSCHEME],
                sum_up_smc[NSCHEME][CYCLE];
        float   sumSMCend[POPSIZE][NSCHEME],
                psumSMCend[NSCHEME],sumNfix[POPSIZE][NSCHEME],
                Nfix[NSCHEME][CYCLE];
        float   psumNfix[NSCHEME],   sumschem_stress[POPSIZE];
        float   N_fixloss;
        float   sumETc[POPSIZE][NSCHEME], psumETc[NSCHEME];

 sumftnss = 0.0f;
for(i=0;i<POPSIZE;i++) {
 ftnss[i] = 0.0f;
 sumLWP[i] = 0.0f;
 sumOversupl[i] = 0.0f;
 sumCapaConstr[i] = 0.0f;
 sumGlobal[i] = 0.0f;
 sumsupply[i] = 0.0f;
 sumEquity[i] = 0.0f;
for(jj = 0; jj < NSCHEME; jj++) { //new
   t_lag = LagC[jj];
   t_lag_adj = (int)(t_lag/Tperiod);
   cycle[i][jj] = (int)(ceil(Calendar/Tperiod) + t_lag_adj);
   sup_wrbd[jj] = (float)(IrriEff*Bflowrate[jj]*Tperiod*3600*24*1000)/(scheme_area[jj]*10000);
}
odds[i] = (float)(Calendar - (Tperiod*(int)floor(Calendar/Tperiod) ) );
Pmutate = (float)(NofMu/LENGTH);      //new
for(tt = 0; tt < STAGE; tt++){
        if(tt == 0)odd2[i][tt] = (float)( (stage_day[tt] - ((int)floor(stage_day[tt]/Tperiod))*Tperiod) );
        else odd2[i][tt] = (float)( (odd2[i][tt-1] + stage_day[tt] -
                        ((int)floor((odd2[i][tt1]+stage_day[tt])/Tperiod))*Tperiod) );
}
 // transfer array!
k = 0;
```

```
        for(jj = 0; jj < NSCHEME; jj++){
                t_lag  = LagC[jj];          //new
          t_lag_adj = (int)(t_lag/Tperiod);
                for(ii     = 0; ii< cycle[i][jj]; ii++){  // new changed from cycle[i][jj] to CYCLE
                        if(ii < t_lag_adj) IFLAG[jj][ii] = 0;
                        if(ii >=  t_lag_adj) {
                                IFLAG[jj][ii] = (int)(initial_pop[i][k]);
                                k++;
                        }
                }
        }
for(jj = 0; jj <NSCHEME; jj++) { //Main jj Loop
         sumX[i][jj] = 0.0f;
         sumEa[i][jj] = 0.0f;
         sumETc[i][jj] = 0.0f;
         suminfil[i][jj] = 0.0f;
         sumDie[i][jj] = 0.0f;
         sumO[i][jj] = 0.0f;
         sumsmc_rz[i][jj] = 0.0f; //new
         cum_stress[i][jj] = 0.0f;//new
         sumIrr_day[i][jj] = 0.0f; //new
         sumSMCend[i][jj] = 0.0f; //new
         sumNfix[i][jj] = 0.0f;      //new

        tt = 0;
        sumcycle = 0.0f;
        countday = (float)(stage_day[0]);
        t_lag = LagC[jj];
        t_lag_adj = (int)(t_lag/Tperiod);
for(ii = 0; ii<cycle[i][jj]; ii++){// Sub LOOP for ii
        if(ii <  t_lag_adj) IFLAG[jj][ii] = 0;
        if(ii >= (t_lag+Calendar)/Tperiod) IFLAG[jj][ii] = 0; //very new
        //new transfer ETc from (mm/period) to be (mm/day) under standard condition
        if(ii < t_lag_adj) {
                ETc[jj][ii] = 0.0f;
                ER[ii] = 0.0f;  //new
        }
        else if(ii >= t_lag_adj) {
        sumcycle = (float)(sumcycle + Tperiod);
        if(sumcycle <= countday){ //new adjust
                ETc[jj][ii] = (float)( ( input_ETc[jj][tt]/stage_day[tt]) *Tperiod);
                ER[ii] = (float)( (Eff_rain[tt]/stage_day[tt]) *Tperiod);
        }
        else{
        if(stagenum[tt] != STAGE){
                        ETc[jj][ii] = (float)( ( (input_ETc[jj][tt])/stage_day[tt]) *odd2[i][tt] +
                        (Tperiod - odd2[i][tt]) * (input_ETc[jj][tt+1])/stage_day[tt+1] ) ; //*odds
                        days

                        ER[ii] = (float)( ( (Eff_rain[tt])/stage_day[tt]) *odd2[i][tt] + (Tperiod -
                        odd2[i][tt]) * (Eff_rain[tt+1])/stage_day[tt+1] ) ; //*odds days
         }
         else {
                        ETc[jj][ii] = (float)( ( (input_ETc[jj][tt])/stage_day[tt]) * odd2[i][tt]);
                        ER[ii] = (float)( ( (Eff_rain[tt])/stage_day[tt]) * odd2[i][tt]);
         }
                tt++;
                countday = countday + stage_day[tt];
```

```
            }
          }
// This is to adjust D_factor  (Allen et al. 1998)
                    if(ETc[jj][ii] > 5.0f){       //new move
                              P[ii] = (float)(D_factr + 0.04*(5 - ETc[jj][ii]));
                              if(P[ii] < 0.1) P[ii] = 0.1f;
                              if(P[ii] > 0.8) P[ii] = 0.8f;
                    }
                    else P[ii] = D_factr;
//Set up smc in every soil strip layer as wilting point of every soil layer
                    FC_layer          = (float)(FC[jj]*layerD*1000);      //new all loop
                    RAM_layer         = (float)((FC[jj] - PWP[jj])*layerD*1000*P[ii]);
                    WP_layer          = FC_layer - RAM_layer;
                    for(nsl = 0; nsl < MaxLyr; nsl++) {
                              smc_layer[nsl][jj][ii-1] = WP_layer;
                    }
// For Kcb adjust and then cal of RD adjust
                    if(ii < t_lag_adj) RD[jj][ii] = 0.0f;
                    else if(ii >= t_lag_adj || smc_rz[jj][ii-1] >= WP_RZ[jj][ii-1]) { //new
                    if(sumcycle < Lini) Kcb[ii] = Kcb_ini; // new (0-24)
                    if(sumcycle >= Lini && sumcycle < Lini+Ldev ){ //(25-49)
                      Kcb[ii] = Kcb_ini + ((sumcycle - Lini)/Lini)*(Kcb_mid - Kcb_ini);
                    }
                    if(sumcycle >= Lini+Ldev && sumcycle < Lini+Ldev+Lmid) Kcb[ii] = Kcb_mid;
                    //(50-79)
                    if(sumcycle >= Lini+Ldev+Lmid  && sumcycle < Calendar){ //new (80-99)
                    Kcb[ii] = Kcb_mid + (( sumcycle - (Lini+Ldev+Lmid))/Llate)*(Kcb_end -
                    Kcb_mid);
                    }
                    if(sumcycle < Lini+Ldev){          //(0-49)
                    RD[jj][ii] = MinRD + (MaxRD - MinRD)*(Kcb[ii] - Kcb_ini)/(Kcb_mid - Kcb_ini);
                              if(RD[jj][ii] > MaxRD) RD[jj][ii] = MaxRD; // new
                    }
                    else{
                              RD[jj][ii] = MaxRD;
                    }
                    }
                    else if(smc_rz[jj][ii-1] < WP_RZ[jj][ii-1]) {
                     if(sumcycle < Lini) Kcb[ii] = Ks[jj][ii]*Kcb_ini;
                     if(sumcycle >= Lini && sumcycle < Lini+Ldev ){
                              Kcb[ii] = Ks[jj][ii]*Kcb_ini + ((sumcycle -
                              Lini)/Lini)*(Ks[jj][ii]*Kcb_mid - Ks[jj][ii]*Kcb_ini);
                    }
                    if(sumcycle >= Lini+Ldev && sumcycle < Lini+Ldev+Lmid) Kcb[ii] =
                    Ks[jj][ii]*Kcb_mid;
                    if(sumcycle >= Lini+Ldev+Lmid  && sumcycle < Calendar){ //new
                              Kcb[ii] = Ks[jj][ii]*Kcb_mid + (( sumcycle -
                              (Lini+Ldev+Lmid))/Llate)*(Ks[jj][ii]*Kcb_end - Ks[jj][ii]*Kcb_mid);
                    }
                    if(sumcycle < Lini+Ldev){
                              RD[jj][ii] = MinRD + (MaxRD - MinRD)*(Kcb[ii] -
                              Ks[jj][ii]*Kcb_ini)/(Ks[jj][ii]*Kcb_mid - Ks[jj][ii]*Kcb_ini);
                              if(RD[jj][ii] > MaxRD) RD[jj][ii] = MaxRD; // new
                    }
                    else{
                              RD[jj][ii] = MaxRD;
                    }
                    }//end of Kcb and RD adjust
```

```
//new for fix loss
                irrig[jj][ii] = sup_wrbd[jj] * IFLAG[jj][ii];
                if(ii == t_lag_adj) {
                        if(IFLAG[jj][ii] == 1) {
                                irrig[jj][ii] = (float)(sup_wrbd[jj] - fixloss) * IFLAG[jj][ii];
                        }
                }
                if(ii > t_lag_adj) {
                        if(IFLAG[jj][ii-1] == 0 && IFLAG[jj][ii] == 1) {
                                irrig[jj][ii] = (float)(sup_wrbd[jj] - fixloss) * IFLAG[jj][ii];
                        }
                }

                //new for sumup of fixloss
                if(ii > t_lag_adj && ii < CYCLE+t_lag_adj){
                        if(IFLAG[jj][ii-1] == 1 && IFLAG[jj][ii] == 0) {
                                countn = 1;
                                Nfix[jj][ii] = (float)(fixloss * countn);
                        }
                        else Nfix[jj][ii] = 0.0f;
                }
                else if(ii == CYCLE+t_lag_adj-1){
                        if(IFLAG[jj][ii-1] == 0 && IFLAG[jj][ii] == 1){
                                countn = 1;
                                Nfix[jj][ii] = (float)(fixloss * countn);
                        }
                        else Nfix[jj][ii] = 0.0f;
                }
}// end of ii loop preparation of all input data
// Now start of Main loop in every cycle
        for(ii = 0; ii < cycle[i][jj]; ii++){// Main LOOP for ii
                / Start to strip soil layer in calculation here ..... new all loop
                nlr_RZ_MAX      = (int)(MaxRD/layerD); //no. of layers at RD=Max
// This is to find out number of layer at root get throuh
// Important!! For example, if NLR_RZ[jj][ii] = 30 that means root has got in layer 0 to 29
// so the layer 30 to 160 is continued to infil
                NLR_RZ[jj][ii]  = (int)(RD[jj][ii]/layerD); // no. of layers at a time period
                RD[jj][ii]                      = (float)(NLR_RZ[jj][ii]*layerD); //adjust RD by layer
                FC_RZ[jj][ii]     = FC[jj]*RD[jj][ii]*1000;
                PWP_RZ[jj][ii]    = PWP[jj]*RD[jj][ii]*1000;
                RAM_RZ[jj][ii]  = (FC[jj] - PWP[jj])*(RD[jj][ii]*1000)*P[ii];
                WP_RZ[jj][ii]     = FC_RZ[jj][ii] - RAM_RZ[jj][ii]; //new for varying p
                if(ii > t_lag_adj){
                        if(NLR_RZ[jj][ii] > NLR_RZ[jj][ii-1]){
                                tsum[jj][ii] = 0.0f;
                                for(nsl = NLR_RZ[jj][ii-1]; nsl < NLR_RZ[jj][ii]; nsl++){
                                        tsum[jj][ii] = tsum[jj][ii] + smc_layer[nsl][jj][ii-1];
                                }
                                smc_rz[jj][ii] = smc_rz[jj][ii-1] + tsum[jj][ii];
                        }
                        else smc_rz[jj][ii] = smc_rz[jj][ii-1];
                }
// This is to sum up smc for global check
        if(ii == t_lag_adj) sum_up_smc[jj][ii] = WP_RZ[jj][ii];
        else sum_up_smc[jj][ii] = smc_rz[jj][ii];
// Now calculate supply[jj][ii] in each time period with IFLAG[jj][ii]
                supply[jj][ii] =  irrig[jj][ii]; //irrig is amount of water supply per time period
                sumsupply[i]   =  sumsupply[i] + supply[jj][ii] * scheme_area[jj] / tot_area;
```

```
//Soil Water Balance Main Equation
                if(ii==t_lag_adj) {
                        Ea[jj][ii] = ETc[jj][ii]; //because initial smc = WP_RZ
                smc_rz[jj][ii]  = (float)(WP_RZ[jj][ii] + supply[jj][ii] - Ea[jj][ii] + ER[ii]);
                if(smc_rz[jj][ii] < WP_RZ[jj][ii]){
                Ea[jj][ii] = (smc_rz[jj][ii] - PWP_RZ[jj][ii])*(ETc[jj][ii])/(WP_RZ[jj][ii]-
                PWP_RZ[jj][ii]);
                smc_rz[jj][ii] = (float)(WP_RZ[jj][ii] + supply[jj][ii] - Ea[jj][ii] + ER[ii]);
                }
                }
                else if(ii > t_lag_adj){
                if(smc_rz[jj][ii] >= WP_RZ[jj][ii]) {
                        Ea[jj][ii] = ETc[jj][ii];
                        smc_rz[jj][ii] = (float)(smc_rz[jj][ii] + supply[jj][ii] - Ea[jj][ii] + ER[ii]);
                }
                        else if(smc_rz[jj][ii] < WP_RZ[jj][ii]){
                        Ea[jj][ii] = (smc_rz[jj][ii] - PWP_RZ[jj][ii])*(ETc[jj][ii])/(WP_RZ[jj][ii]-
                        PWP_RZ[jj][ii]);
                        smc_rz[jj][ii] = (float)(smc_rz[jj][ii] + supply[jj][ii] - Ea[jj][ii] + ER[ii]);
                        }
                }
// end Soil Moisture Balance Main Equation and its adjustment.
                SMCline[jj][ii] = smc_rz[jj][ii]; //for ploting
                // this is when SMC > FC
                infilt = 0.0f;
                if(smc_rz[jj][ii] > FC_RZ[jj][ii]){ //when smc_rz OVER FC_RZ > infil cal. here
                        SMCline[jj][ii]  =  FC_RZ[jj][ii];//for plotting
                infilt = smc_rz[jj][ii] - FC_RZ[jj][ii];
                infiltopup[jj][ii] = infilt; //this is to find how much infiltration top up in this ii
                        smc_rz[jj][ii] =  FC_RZ[jj][ii];
                // this is to put infiltration from layer to layer below
                        for(nsl = NLR_RZ[jj][ii]; nsl < nlr_RZ_MAX; nsl++) {
                                smc_layer[nsl][jj][ii] = smc_layer[nsl][jj][ii-1] + infilt;
                                if(smc_layer[nsl][jj][ii] > FC_layer){
                                        infilt = smc_layer[nsl][jj][ii] - FC_layer;
                                        smc_layer[nsl][jj][ii] = FC_layer;
                                }
                                else {
                                        infilt = 0.0;
                                        break;
                                }
                        }
                        Ovr_supl[jj][ii] = infilt;
                }
                else{
                        for(nsl = 0; nsl < nlr_RZ_MAX; nsl++) {
                        // if there is no infil so, smc_layer[ii] = smc_layer[ii-1] last cycle
                                smc_layer[nsl][jj][ii] = smc_layer[nsl][jj][ii-1];
                        }
                        infiltopup[jj][ii] = 0.0f;
                        Ovr_supl[jj][ii]  = 0.0f;
                }
                sumOversupl[i] = sumOversupl[i] + Ovr_supl[jj][ii];
// This is to compute net_infil this cycle
                if(smc_rz[jj][ii] > FC_RZ[jj][ii])    netinfil[jj][ii] = infiltopup[jj][ii] -
                Ovr_supl[jj][ii];
                else netinfil[jj][ii] = 0.0f;
// This is for plotting smc line if oversupply
```

```
                if(Ovr_supl[jj][ii] > 0.0f) SMCline[jj][ii]  = FC_RZ[jj][ii] + Ovr_supl[jj][ii]; //new
// This is for calculation of Ks when smc_rz < WP_RZ
                if(smc_rz[jj][ii] < WP_RZ[jj][ii]) {
                        Ks[jj][ii] = (FC_RZ[jj][ii] - smc_rz[jj][ii])/(FC_RZ[jj][ii]-WP_RZ[jj][ii]);
                }
                else Ks[jj][ii] = 1;
// This is a penalty function.
                if(smc_rz[jj][ii] < WP_RZ[jj][ii]) {
                        // Now must define once again incase if smc < pwp
                        LWP[jj][ii] = (float)(WP_RZ[jj][ii] - smc_rz[jj][ii]);//this if for scarcity
                        Equity[jj][ii] = (float)pow((R1*LWP[jj][ii]),2);
// This is for plotting of SMCline if smc < WP
                SMCline[jj][ii] = WP_RZ[jj][ii] - LWP[jj][ii];
                }
                else {
                        LWP[jj][ii] = 0.0f;
                        Equity[jj][ii] = 0.0f;
                }
                sumLWP[i]    = sumLWP[i] + LWP[jj][ii];
                sumEquity[i] = sumEquity[i] + Equity[jj][ii];
                sumsmc_rz[i][jj]  = sumsmc_rz[i][jj] + sum_up_smc[jj][ii];//this is for global check
                sumX[i][jj]      = sumX[i][jj]     + supply[jj][ii];       //sum supply exclude fixloss
                sumETc[i][jj]  = sumETc[i][jj]  + ETc[jj][ii];
                sumEa[i][jj]     = sumEa[i][jj] + Ea[jj][ii];   //sum actual evapotranspiration
                sumSMCend[i][jj]  = sumSMCend[i][jj] + smc_rz[jj][ii];
                sumO[i][jj]  = sumO[i][jj]      + Ovr_supl[jj][ii];  //sum oversupply above RDmax
                cum_stress[i][jj] = cum_stress[i][jj]+ LWP[jj][ii];
                sumNfix[i][jj]   = sumNfix[i][jj]   + Nfix[jj][ii];  //sum fixloss
                suminfil[i][jj] = suminfil[i][jj]  + infiltopup[jj][ii]; //infil if smc > FC include ovrsup
                GlobalSMC[i][jj]  = sumsmc_rz[i][jj] + sumX[i][jj] - sumEa[i][jj] -
                sumSMCend[i][jj] - suminfil[i][jj];
                sumGlobal[i]     = sumGlobal[i] + GlobalSMC[i][jj];
                sumIrr_day[i][jj] = sumIrr_day[i][jj] + IFLAG[jj][ii];   //new
          }//end of ii loop
}//end of jj scheme
 // this is canal capacity constraint
 for(ii=0; ii<CYCLE; ii++){            //new
        sumBflow[ii] = 0.0f;
        for(jj = 0; jj <NSCHEME; jj++) {
                branchGO[ii][jj]  = Bflowrate[jj]*IFLAG[jj][ii];
                sumBflow[ii]     = sumBflow[ii] + branchGO[ii][jj];
        }
        if(sumBflow[ii] > main_capa) {
        capaConstr[i][ii] = (float)(R2*(sumBflow[ii] - main_capa));
        }
        else capaConstr[i][ii] = 0.0f;
        sumCapaConstr[i] = sumCapaConstr[i] + (float)(pow(capaConstr[i][ii],2));
}//end of ii in capacity constraint loop
        // Objective function is Here
        ftnss[i] = (float)(sumsupply[i] + sumEquity[i] + sumCapaConstr[i]);
        sumftnss = sumftnss + ftnss[i];
        if(bestfit  > ftnss[i]){
                bestfit  = ftnss[i];
                psumLWP  = sumLWP[i];
                psumOVR  = sumOversupl[i];
                psumCapa  = sumCapaConstr[i];
                psumsupply= sumsupply[i];
                psumEquity= sumEquity[i];
```

```
                for(jj = 0; jj<NSCHEME; jj++) {
                        psumX[jj] = sumX[i][jj];
                        psumEa[jj] = sumEa[i][jj];
                        psumETc[jj] = sumETc[i][jj];
                        psumsmc_rz[jj] = sumsmc_rz[i][jj];
                        psumSMCend[jj] = sumSMCend[i][jj];
                        psumDie[jj] = sumDie[i][jj];
                        psumNfix[jj] = sumNfix[i][jj];
                        psumO[jj] = sumO[i][jj];
                        pstress[jj] = cum_stress[i][jj];
                        pGlobalSMC[jj] = GlobalSMC[i][jj];
                        psumIrr_day[jj] = sumIrr_day[i][jj];
                        psuminfil[jj] = suminfil[i][jj];
                        psumnetinfil[jj] = sumnetinfil[i][jj];
            for(ii=0; ii<cycle[i][jj]; ii++) {
                pFlag[jj][ii] = IFLAG[jj][ii];
                pRD[jj][ii] = RD[jj][ii];
                pEa[jj][ii] = Ea[jj][ii];
                ptsum[jj][ii] = tsum[jj][ii];
                pinfiltopup[jj][ii] = infiltopup[jj][ii];
                psupply[jj][ii] = supply[jj][ii];
                pdemand[jj][ii] = demand[jj][ii];
                psmc_rz[jj][ii] = smc_rz[jj][ii];
                pSMCline[jj][ii] = SMCline[jj][ii];
                pOvr_supl[jj][ii] = Ovr_supl[jj][ii];
                pLWP[jj][ii] = LWP[jj][ii];
                pcapaConstr[ii] = capaConstr[i][ii];
                }
                }
                }
}//end of i
if (gen == MaxGEN-1 || DELTA_ftnss <= DELTA ) {
index = 999;
     for(jj = 0; jj<NSCHEME; jj++) {
        N_fixloss          = (float)(psumNfix[jj]/fixloss);
for(ii=0; ii<CYCLE; ii++) fprintf(f1,"%8d",pFlag[jj][ii]);
fprintf(f1, " Irri_day  %9.3f sumFixloss  %9.3f\n", psumIrr_day[jj], psumNfix[jj]);
printf(f1,"FC     ");
for(ii=0; ii<CYCLE; ii++) fprintf(f1,"%8.3f",FC_RZ[jj][ii]); fprintf(f1,"\n");
        fprintf(f1,"supply  ");
        for(ii=0; ii<CYCLE; ii++) fprintf(f1,"%8.3f",psupply[jj][ii]); fprintf(f1,"\n");
        fprintf(f1,"Ea      ");
        for(ii=0; ii<CYCLE; ii++) fprintf(f1,"%8.3f",pEa[jj][ii]); fprintf(f1,"\n");
        fprintf(f1,"smc_rz  ");
        for(ii=0; ii<CYCLE; ii++) fprintf(f1,"%8.3f",psmc_rz[jj][ii]); fprintf(f1,"\n");
        fprintf(f1,"SMC_line ");
        for(ii=0; ii<CYCLE; ii++) fprintf(f1,"%8.3f",pSMCline[jj][ii]);          fprintf(f1,"\n");
        fprintf(f1,"WP      ");
        for(ii=0; ii<CYCLE; ii++) fprintf(f1,"%8.3f",WP_RZ[jj][ii]); fprintf(f1,"\n");
        fprintf(f1,"PWP     ");
        for(ii=0; ii<CYCLE; ii++) fprintf(f1,"%8.3f",PWP_RZ[jj][ii]); fprintf(f1,"\n");
        fprintf(f1,"Ovr_supl ");
        for(ii=0; ii<CYCLE; ii++) fprintf(f1,"%8.3f",pOvr_supl[jj][ii]); fprintf(f1,"\n");
        fprintf(f1,"BelowWP ");
        for(ii=0; ii<CYCLE; ii++) fprintf(f1,"%8.3f",pLWP[jj][ii]);
        fprintf(f1, " cum_stress %9.3f\n", pstress[jj]);
        fprintf(f1,"Global %9.3f\n", pGlobalSMC[jj]);
        fprintf(f1,"SMCstr %9.3f\n", psumsmc_rz[jj]);
```

```
        fprintf(f1,"Irriga %9.3f\n", psumX[jj]);
        fprintf(f1,"ETc   %9.3f\n", psumETc[jj]);
        fprintf(f1,"Ea__  %9.3f\n", psumEa[jj]);
        fprintf(f1,"SMCend %9.3f\n", psumSMCend[jj]);
        fprintf(f1,"Ovrsup %9.3f\n", psumO[jj]);
        fprintf(f1,"infilt %9.3f\n", psuminfil[jj]);
        fprintf(f1,"stress %9.3f\n", pstress[jj]);
        fprintf(f1,"fixlos %9.3f\n", psumNfix[jj]);
}// end of scheme
fprintf(f1,"\n");
fprintf(f1," %1.0f  %4.2f  %4.3f  %3d  R1%2d R2%2d", Tperiod, PXOVER, NofMu, gen, R1, R2);
fprintf(f1," %7.3f   %7.3f   %6.3f   %6.3f\n",
        ftnss[POPSIZE], psumsupply , psumEquity,  psumCapa);
}// end of if loop

  avgftnss = sumftnss/POPSIZE;
  minftnss  = ftnss[0];
  for(i=0; i<POPSIZE; i++) {
    if(ftnss[i]  < minftnss)   minftnss  = ftnss[i];
          if(ftnss[i]  < ftnss[POPSIZE]){
                ftnss[POPSIZE] = ftnss[i];
          }
  }
fprintf(f1,"%3d %10.3f %7.3f\n",gen+1,ftnss[POPSIZE],bestfit);
}
/*************************************/
void Tournament() {
        int  ii,jj,j,mem;
        for(mem=0; mem < POPSIZE; mem++){
                ii = RandInt(0,POPSIZE-1);
                jj = RandInt(0,POPSIZE-1);
                if(ftnss[ii] <= ftnss[jj]){
                        for(j=0; j<LENGTH; j++){
                                mated_pop[mem][j] = initial_pop[ii][j];
                        }
                }
                if(ftnss[ii] > ftnss[jj]){
                        for(j=0; j<LENGTH; j++){
                                mated_pop[mem][j] = initial_pop[jj][j];
                        }
                }
}
/************/
// Routine to crossover population
void XOVER(int i1, int i2) {
        int j,kk;
        kk = RandInt(1,LENGTH-1);
        for(j=kk;j<LENGTH;j++)  {
//   SwapIntValues(&mated_pop[i1][j],&mated_pop[i2][j]);
         SwapRealValues(&mated_pop[i1][j],&mated_pop[i2][j]);
   }
}
/*************************************/
void UNIXOVER(int i1, int i2){
        int j,kk;
        for(j=0;j<LENGTH;j++)  {
                kk= rand()&01;
//      if(kk==1) SwapIntValues(&mated_pop[i1][j],&mated_pop[i2][j]);
```

```
                if(kk==1) SwapRealValues(&mated_pop[i1][j],&mated_pop[i2][j]);
    }
}
/*************************************/
// Routine to crossover population, 2-site
void TPXOVER(int i1, int i2){
        int     j,kk1,kk2;
        kk1 = RandInt(1,LENGTH-1);
        kk2 = RandInt(1,LENGTH-1); // this is for 2-site XOVER
        if (kk1>kk2) SwapIntValues(&kk1,&kk2); // this is to swap 2-site if the later is less
        for(j=kk1;j<kk2;j++)  {
//      SwapIntValues(&mated_pop[i1][j],&mated_pop[i2][j]);
                SwapRealValues(&mated_pop[i1][j],&mated_pop[i2][j]);
    }
}
/*************************************/
void SelectMate() {
int  i,j,mem, num_select,one;
float Prob_X;
num_select = 0;
for (mem = 0; mem< POPSIZE ; mem++) {
        Prob_X = rand()%1000/1000.0f;
        if (Prob_X < PXOVER){
        ++num_select;
        if(num_select%2 == 0){
                TPXOVER(one,mem);  // for testing 1-site use XOVER, for 2-site use TPXOVER
        }
        else one = mem;
}
        }
/*******************************/
void UniSelectMate() {
int   i, j, mem, num_select,one;
float Prob_X;
num_select = 0;
        for (mem = 0; mem< POPSIZE ; mem++) {
                Prob_X = rand()%1000/1000.0f;
                if (Prob_X < PXOVER){
                        ++num_select;
                        if(num_select%2 == 0){
                                UNIXOVER(one,mem);
                        }
                        else one = mem;
                }
        }
}
/*****************************************/
// this is to random probability of mutation
void mutatepop() {
int i,j;
double r;
        for (i=0;i<POPSIZE;i++) {
        for (j=0;j<LENGTH;j++) {
        r = rand()%1000/1000.0;
                if(r<Pmutate){
                        if(mated_pop[i][j] == 0)  mated_pop[i][j] = 1;
                        else mated_pop[i][j] = 0;
                 }
```

```
        }
        for(j=0;j<LENGTH;j++)   initial_pop[i][j] = mated_pop[i][j];
        }
/*************************************/
// Routine to print error message
void nrerror(char *messg) {
        puts(messg) ;
        exit(1) ;
}
/*************************************/
// Routine to swop value
void SwapIntValues(int *x, int *y) {
        int temp;
        temp = *x;
        *x=*y;
        *y=temp;
}
/*************************************/
// Routine to swop real value
void SwapRealValues(float *xx, float*yy) {
        float temp;
        temp = *xx;
        *xx = *yy;
        *yy = temp;
}
/*************************************/
```

//GA46A.c **Program to formulate GAs for water scheduling problem**

```
                    //ZERO-1 CRITERIA for a more complex system (chapter 7)
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <process.h>
#include <time.h>

#define Tperiod         1.0     // input time period in considered
#define POPSIZE         100     // input population size
#define INTERVAL        300     // input interval of generation to recheck of improvement fitness
#define DELTA  0.0001   // input minimum improvement of fitness after INTERVAL generation
#define IrriEff         0.837 //Irri efficiency of secondaries(0.837), of Teriaries(0.837)
#define  fixloss        1.5     // fixloss of each teriary canal
#define PercentFlow     100     // input percent of main canal flow
#define NUMCROP         1       // input number of crop
#define MAXSUBLAT       4       // input maximum tertiary canals in a secondary
#define NSCHEME         4       // input total scheme numbers
#define SumSubLat       16      // input total tertiaries of the system
#define Calendar        100     // input maximum crop growth period
#define STAGE           100     // input stage of input data
#define CYCLE           115     // input total time period calendar + max. straggering days
#define LENGTH          CYCLE*MAXSUBLAT  // input chromosome length of a secondary
#define inc             1.0     // input increment to mutate
#define  layerD         0.005   // input a root zone soil layer for water balance in meter
#define  tDepth         1.0     // input total depth of soil layer in meter
#define MaxLyr          200     // input maximum soil layer in consider

int     RandInt (int, int);
float   RandReal (float,float);
void    Read_inputfiles();
void    SwapIntValues(int *, int *);
void    SwapRealValues(float *, float *);
void    nrerror(char *messg) ;
void    initial();
void    objective(int);
void    SelectMate();
void    UniSelectMate();
void    mutatepop();
void    Tournament();

int     MaxGEN, R1, R2, R3, t_lag;
float   NofMu, PXOVER;
int     i ,tt, index, stage_day[STAGE], num_sublat[NSCHEME];
int     g, stage_num[STAGE],IFLAG[NSCHEME][MAXSUBLAT][CYCLE];
int     pFlag[NSCHEME][MAXSUBLAT][CYCLE]; //new
int     IFLAG_B[NSCHEME][CYCLE], pFlag_B[NSCHEME][CYCLE]; // new
float   sumIFLAG[NSCHEME][CYCLE]; //new
float   countday;
float   mated[POPSIZE],bestfit, Pmutate,ftnss[POPSIZE+1], DELTA_ftnss, OLD_minftnss;
float   initial_pop[POPSIZE][NSCHEME][LENGTH],
        mated_pop[POPSIZE][NSCHEME][LENGTH];
float   MinRD, MaxRD, Lini, Ldev, Lmid, Llate, Kcb_ini, Kcb_mid, Kcb_end, Kcb[CYCLE];
float   RootD[STAGE], D_factr, FC[NSCHEME][MAXSUBLAT],
PWP[NSCHEME][MAXSUBLAT];
```

```
float     RAM[NSCHEME][MAXSUBLAT][CYCLE],
          pRAM[NSCHEME][MAXSUBLAT][CYCLE];
float     Bcapa[NSCHEME],loss_fct[NSCHEME],
input_ETc[NSCHEME][MAXSUBLAT][STAGE], main_capa;
float     Scapa[NSCHEME][MAXSUBLAT], scheme_area[NSCHEME], Eff_rain[STAGE];
float     subLat_area[NSCHEME][MAXSUBLAT], TotalAREA;
int       LagC[NSCHEME][MAXSUBLAT];

FILE *f1,*f2,*f3 ;

//MAIN PROGRAM
void main (void) {
          int gen;
          clock_t start, finish;
          float  duration, TotalTime ;
   TotalTime = 0.0f;
          if(!(f1 = fopen("ga46.out","w"))) {
                    printf("can't open file output file \n");
                    exit(1);
          }
   Read_inputfiles();
          srand(2);

          MaxGEN          =       2500;
          PXOVER          =       0.9f;
          NofMu           =       0.05f;
          R1              =       1;
          R2              =       2500;
          R3              =       5000;

//        R1 =  1;
//do{
//        R2      =       12500;
//do{
//        R3      =       5000;
//do{

/*        NofMu  = 0.01f;
do{
   PXOVER      = 0.5f;
do{ */

   initial();
   index = 0;
          OLD_minftnss    = 1000.0f;
          DELTA_ftnss     = 1000.0f;
          bestfit         = 10000000000.0f;
          ftnss[POPSIZE]  = 100000000000.0f;

 start = clock();
 for(gen = 0; gen <= MaxGEN; gen++) {
          objective(gen); if(index == 999) break;
          Tournament();
//        SelectMate();
          UniSelectMate();
   mutatepop();
   if (!((gen+1)%INTERVAL)){
     DELTA_ftnss = (float)fabs((double)((OLD_minftnss - ftnss[POPSIZE])/ftnss[POPSIZE]));
```

```
      OLD_minftnss = ftnss[POPSIZE];
    }
  } //End of Generation Loop
  finish = clock();
  duration = (float)(finish - start) / CLOCKS_PER_SEC;
  TotalTime = (float)(TotalTime + duration);
  printf(" Time = %5.2f sec\n", duration);
  fprintf(f1,"New Pmu Time = %5.2f sec \n\n", duration);
/*PXOVER = (float)(PXOVER + 0.05f);
}while (PXOVER <= 1.01f);
  NofMu = (float)(NofMu + 0.02);
}while (NofMu  <= 0.14); */
//  R3 = R3 + 2500;
//}while (R3 <= 10001);
//  R2 = R2 + 2500;
//}while (R2 <= 15001);
//  R1 = R1 + 2;
//}while (R1 <= 6);
  (void) fclose (f1);
  printf("End of run  ENJOY!!!! ");
} //MAIN PROGRAM END
/*************************/
void Read_inputfiles() {
        int  ii, jj, kk, cc, num_schemes, scheme_num[NSCHEME];
        int  scheme_num1[NSCHEME], scheme_num2[NSCHEME], scheme_num3[NSCHEME],
        scheme_num4[NSCHEME], scheme_num5[NSCHEME];
        int  num_crop, crop_nmbr2,crop_period[NUMCROP];
        int  sublat[MAXSUBLAT], Nstage, crop_nmbr;
        float crop_area[NUMCROP][NSCHEME];
        char  title1[80],title2[80],title3[80], title5[80],title6[80], title7[80];
        char  title8[80], title9[80], title10[80], title11[80];
        char  schedulefile[20];
        if(!(f2 = fopen("Proj46A.lst","r"))) {     //for input46A.dat
                nrerror(" Can not open project file") ;
        }
// start reading file here
        fgets(schedulefile, 20, f2) ;
        for(ii = 0; ii < 20; ii++) {if(schedulefile[ii] == '\n') schedulefile[ii] = '\0' ;
        (void) fclose(f2) ;
// now read water scheduling data
        if(!(f2 = fopen(schedulefile,"r")))  nrerror(" Unable to open schedulefile") ;
// Canal data file start here **********************
   fgets(title1, 80, f2);        // Canal data title
   fscanf(f2,"%d",&num_schemes);  // scan number of schemes here
   for(jj=0;jj<num_schemes;jj++) fscanf(f2,"%d ", &scheme_num[jj]);
        for(jj = 0; jj < NSCHEME; jj++) fscanf(f2,"%d", &num_sublat[jj]);
        fscanf(f2,"\n");
        fscanf(f2,"%d",&num_crop);  // number of crop (how many crops in this system
        for(jj = 0; jj < NSCHEME; jj++) {
        fscanf(f2,"%d",&scheme_num1[jj]);
        for(kk = 0; kk < num_sublat[jj]; kk++){
                fscanf(f2,"%d", &sublat[kk]);
                for (cc = 0; cc < num_crop; cc++){
                        fscanf(f2,"%f",&crop_area[cc][jj]);
                }
        }
        }
        fscanf(f2,"\n");
```

```
        TotalAREA = 0.0f;
for(jj = 0; jj < NSCHEME; jj++) {
                scheme_area[jj] = 0.0f;
        for(kk = 0; kk<num_sublat[jj]; kk++){
                subLat_area[jj][kk] = 0.0f;
                        for (cc = 0; cc < num_crop; cc++){
                                subLat_area[jj][kk] = subLat_area[jj][kk] + crop_area[cc][jj];
                        }
                        scheme_area[jj] = scheme_area[jj] + subLat_area[jj][kk];
        }
        TotalAREA = TotalAREA + scheme_area[jj];
}
  fgets(title2, 80, f2);        // Main & lateral canals title
        fscanf(f2,"%f ", &main_capa) ; // scan main canal supply
  // scan branch canal supply
        for(jj = 0; jj < num_schemes; jj++) fscanf(f2,"%f ", &Bcapa[jj]) ;
        fscanf(f2,"\n");
  //scan sub-lateral capa
        fgets(title3, 80, f2);
        for(jj = 0; jj < NSCHEME; jj++) {
                fscanf(f2,"%d ", &scheme_num2[jj]);
                for(kk = 0; kk < num_sublat[jj]; kk++) {
                        fscanf(f2,"%f ", &Scapa[jj][kk]);
                }
        }
  fscanf(f2,"\n");
  // Soil data file start here ******************
  fgets(title5, 80, f2);        // LagC
        for(jj = 0; jj < num_schemes; jj++) {
                fscanf(f2,"%d ", &scheme_num3[jj]);
                for(kk = 0; kk < MAXSUBLAT; kk++){
                 fscanf(f2,"%d ", &LagC[jj][kk]) ;
                }
        }
        fscanf(f2,"\n");
        fgets(title6, 80, f2);        // field capacity title
        for(jj = 0; jj < num_schemes; jj++) {
                fscanf(f2,"%d ", &scheme_num4[jj]);
        for(kk = 0; kk < MAXSUBLAT; kk++){
                 fscanf(f2,"%f ", &FC[jj][kk]) ;    // scan FC
                }
        }
  fscanf(f2,"\n");
  fgets(title7, 80, f2);        // permanent wilting point title
        for(jj = 0; jj < num_schemes; jj++) {
                fscanf(f2,"%d ", &scheme_num5[jj]);
        for(kk = 0; kk < MAXSUBLAT; kk++){
                 fscanf(f2,"%f ", &PWP[jj][kk]) ;   // scan PWP
                }
        }
  fscanf(f2,"\n");
        // Crop Data File start here  ******************

  fgets(title8, 80, f2);                                        // Detail crop datas
        fscanf(f2,"%3d",&crop_nmbr);
        for (cc = 0; cc < crop_nmbr; cc++){
                fscanf(f2,"%3d",&crop_period[cc]);//Days of Max. crop calender (100)
        }
```

```
        fscanf(f2,"%3d",&crop_nmbr2);     // crop number (1)
        fscanf(f2,"%3d ",&Nstage);                    /************************/
        for(tt = 0; tt < Nstage; tt++){
                fscanf(f2,"%3d",&stage_num[tt]);          // total stage number = 1,2,3....100
        }
        fscanf(f2,"\n");
        for(tt = 0; tt < Nstage; tt++){
                fscanf(f2,"%2d",&stage_day[tt]);          // number of days in each stage (1)
        }
        fscanf(f2,"\n");
  for(jj = 0; jj <NSCHEME; jj++) {
        for(kk = 0; kk < MAXSUBLAT; kk++){
                 for(tt = 0; tt < Nstage; tt++){
                        fscanf(f2,"%f", &input_ETc[jj][kk][tt]);      //input_ETc[jj][kk][tt]
                 }
                }
        }
  fscanf(f2,"\n");
        for(tt = 0; tt <Nstage; tt++){                  // Effective rainfall
          fscanf(f2,"%f", &Eff_rain[tt]) ;
        }
  fscanf(f2,"\n");
  fgets(title9,80,f2);   // Length stage title
        fscanf(f2,"%f %f %f %f",&Lini, &Ldev, &Lmid, &Llate);
  fscanf(f2,"\n");
        fgets(title10,80,f2);   // Kcb title
        fscanf(f2,"%f %f %f",&Kcb_ini, &Kcb_mid, &Kcb_end);
  fscanf(f2,"\n");
  fgets(title11, 80, f2); // rooting depth title
        fscanf(f2,"%f %f", &MinRD, &MaxRD);
        fscanf(f2,"%f",&D_factr); // Defraction factors
  (void)fclose(f2);
}//end Readinput file
/*********/
float RandReal (float lower,float upper) {
        float val;
        float nb = ( upper - lower );
        val =  (float)(rand()%1000/1000.0*nb + lower);
        return ((float)val);
}
/*********/
int RandInt (int lower,int upper) {
        int val;
        int nb = (upper - lower );
        val =  (rand()%nb + lower);
        return val;
}
/*********/
void initial() {
  int i, j, k;
  float nd = RAND_MAX;
  for(i=0;i<POPSIZE;i++) {
          for(j=0; j< NSCHEME; j++){
                  for(k=0; k<LENGTH; k++){
                          initial_pop[i][j][k] = (float)(rand()&01);
                  }
          }
  }
```

```
/************************/
void objective(int gen){
int     i, j, k, ii, kk, cycle[POPSIZE][NSCHEME][MAXSUBLAT];
float   sumftnss, minftnss, avgftnss;
float   odds[POPSIZE][NSCHEME], odd2[NSCHEME][STAGE],
        sumcycle[NSCHEME][MAXSUBLAT];
float   Ovr_supl[NSCHEME][MAXSUBLAT][CYCLE],
        pOvr_supl[NSCHEME][MAXSUBLAT][CYCLE];
float   LWP[NSCHEME][MAXSUBLAT][CYCLE],
        LWP[NSCHEME][MAXSUBLAT][CYCLE];
float   S_latGO[LENGTH][MAXSUBLAT], pETc[NSCHEME][MAXSUBLAT][CYCLE];
float   ETc[NSCHEME][MAXSUBLAT][CYCLE],   sup_wrbd[NSCHEME][MAXSUBLAT];
float   supply[NSCHEME][MAXSUBLAT][CYCLE],
        psupply[NSCHEME][MAXSUBLAT][CYCLE];
float   demand[NSCHEME][MAXSUBLAT][CYCLE],
        pdemand[NSCHEME][MAXSUBLAT][CYCLE];
float   psumOVR, psumLWP, psumBcapa, RD[NSCHEME][MAXSUBLAT][CYCLE];
float   sumBcapaCon[POPSIZE], sumLWP[POPSIZE];
float   capaConstr[NSCHEME][CYCLE], pcapaConstr[NSCHEME][CYCLE];
float   lobalSMC[POPSIZE][NSCHEME][MAXSUBLAT],
        pGlobalSMC[NSCHEME][MAXSUBLAT];
float   sumX[POPSIZE][NSCHEME][MAXSUBLAT],
        sumD[POPSIZE][NSCHEME][MAXSUBLAT];
float   sumD[NSCHEME][MAXSUBLAT], psumX[NSCHEME][MAXSUBLAT];
float   sumsmc_rz[POPSIZE][NSCHEME][MAXSUBLAT],
        psumsmc_rz[NSCHEME][MAXSUBLAT];
float   mc_rz[NSCHEME][MAXSUBLAT][CYCLE],
        psmc_rz[NSCHEME][MAXSUBLAT][CYCLE];
float   smO[POPSIZE][NSCHEME][MAXSUBLAT], sumGlobal[POPSIZE],
        sumsupply[POPSIZE];
float   nCapaCon[CYCLE], sumMCapaCon[POPSIZE], psumMCapaCon;
float   umTertiaryFlow[NSCHEME][CYCLE];
float   SMCline[NSCHEME][MAXSUBLAT][CYCLE],
        pSMCline[NSCHEME][MAXSUBLAT][CYCLE];
float   sumEquity[POPSIZE], psumsupply, psumEquity,
        Equity[NSCHEME][MAXSUBLAT][CYCLE];
float   WPscheme[POPSIZE][NSCHEME][MAXSUBLAT],
        pLWPscheme[NSCHEME][MAXSUBLAT];
float   sumSecondaryFlow[CYCLE], sumOversupl[POPSIZE];
float   P[CYCLE], Ks[NSCHEME][MAXSUBLAT][CYCLE];
int     NLR_RZ[NSCHEME][MAXSUBLAT][CYCLE], nlr_RZ_MAX, nsl, countn;  //new
float   C_layer, RAM_layer, WP_layer, RAM_RZ[NSCHEME][MAXSUBLAT][CYCLE]; //new
float   FC_RZ[NSCHEME][MAXSUBLAT][CYCLE],
        PWP_RZ[NSCHEME][MAXSUBLAT][CYCLE],
        WP_RZ[NSCHEME][MAXSUBLAT][CYCLE];//new
float   smc_layer[200][NSCHEME][MAXSUBLAT][CYCLE],
        tsum[NSCHEME][MAXSUBLAT][CYCLE];
float   Ea[NSCHEME][MAXSUBLAT][CYCLE], infilt ; //new
float   pstress[NSCHEME][MAXSUBLAT], cum_stress[POPSIZE][NSCHEME][MAXSUBLAT];
float   ER[CYCLE], irrig[NSCHEME][MAXSUBLAT][CYCLE],
        Nfix[NSCHEME][MAXSUBLAT][CYCLE];
float   sum_up_smc[NSCHEME][MAXSUBLAT][CYCLE];
        float    infiltopup[NSCHEME][MAXSUBLAT][CYCLE];
float   sumNfix[POPSIZE][NSCHEME][MAXSUBLAT],
        sumEa[POPSIZE][NSCHEME][MAXSUBLAT];
float   suminfil[POPSIZE][NSCHEME][MAXSUBLAT],
        sumSMCend[POPSIZE][NSCHEME][MAXSUBLAT];
```

```
float     sumIrr_day[POPSIZE][NSCHEME][MAXSUBLAT],
          psuminfil[NSCHEME][MAXSUBLAT];
float     psumEa[NSCHEME][MAXSUBLAT],psumSMCend[NSCHEME][MAXSUBLAT];
          float     psumNfix[NSCHEME][MAXSUBLAT], psumO[NSCHEME][MAXSUBLAT];
float     psumIrr_day[NSCHEME][MAXSUBLAT],pEa[NSCHEME][MAXSUBLAT][CYCLE] ;
float     ptsum[NSCHEME][MAXSUBLAT][CYCLE], N_fixloss, B_GO[CYCLE][NSCHEME];
float     psumETc[NSCHEME][MAXSUBLAT], sumETc[POPSIZE][NSCHEME][MAXSUBLAT];
          sumftnss = 0.0f;
for(i=0;i<POPSIZE;i++)  {
          ftnss[i]            = 0.0f;
          sumLWP[i]           = 0.0f;
          sumOversupl[i]      = 0.0f;
          sumBcapaCon[i]      = 0.0f;
          sumMCapaCon[i]            = 0.0f;
          sumsupply[i]        = 0.0f;
          sumGlobal[i]        = 0.0f;
          sumEquity[i]        = 0.0f;

  /****** START to Transfer Array here ******/
          for(j = 0; j < NSCHEME; j++) { //new
          for(k = 0; k < num_sublat[j]; k++) {
                              t_lag = LagC[j][k];
                              cycle[i][j][k] = (int)(ceil(Calendar/Tperiod)+t_lag);
          }
          odds[i][j]   = (float)(Calendar - (Tperiod*(int)floor(Calendar/Tperiod)) );
          Pmutate      = (float)(NofMu/(LENGTH*NSCHEME));
          }// end LOOP J1
          // every stage of month it has odd2 that occours when devide stageday with wrbd
          for(j = 0; j<NSCHEME; j++){ // LOOP J2
          for(tt = 0; tt<STAGE; tt++){
          if(tt == 0) {
                    odd2[j][tt] = (float)( stage_day[tt] - ((int)floor(stage_day[tt]/Tperiod))*Tperiod );
                    // odd2 in first stage = (30 - (floor(30/2.25)*2.25) = 0.75 days
          }
          else{
                    odd2[j][tt] = (float)(odd2[j][tt-1] + stage_day[tt] - ((int)floor((odd2[j][tt-
                    1]+stage_day[tt])/Tperiod))*Tperiod );
          // later stage continuing count odd2 from the first stage =  0.75 days + 31 - ( floor((0.75 +
          31)/2.25)*2.25 = 0.25 days thus summation of odds become 5 in stage 2
          }
          }
          } // end  J2
    for(j = 0; j<NSCHEME; j++) {
          kk = 0;
          for(k = 0; k<num_sublat[j]; k++){
                    t_lag = LagC[j][k]; //new
                    for(ii = 0; ii<cycle[i][j][k]; ii++){    // cycles in each sub_lateral
                    IFLAG[j][k][ii] = 0;
                    if(ii >= t_lag && ii < t_lag+Calendar) IFLAG[j][k][ii] = (int)(initial_pop[i][j][kk]);
                    kk++;
                    }
                    }
          }

          // Transfer Array end
          for(j = 0; j<NSCHEME; j++) { //MAIN SCHEME LOOP for j
          for(k = 0; k < num_sublat[j]; k++) { // MAIN LOOP for k
            sumX[i][j][k]             = 0.0f;
```

```
                sumEa[i][j][k]     = 0.0f;
                sumETc[i][j][k]   = 0.0f;
                suminfil[i][j][k]  = 0.0f;
                sumO[i][j][k]      = 0.0f;
                sumsmc_rz[i][j][k] = 0.0f;
                cum_stress[i][j][k] = 0.0f;
                sumIrr_day[i][j][k] = 0.0f;
                sumSMCend[i][j][k] = 0.0f;
                sumNfix[i][j][k] = 0.0f;
                LWPscheme[i][j][k]  = 0.0f;
 tt = 0;
 sumcycle[j][k] = 0.0f;
     countday =  (float)(stage_day[0]);
     t_lag = LagC[j][k]; // new move
for(ii = 0; ii< cycle[i][j][k]; ii++){   // sub ii loop for preparing data is HERE
    if(ii <  t_lag) IFLAG[j][k][ii] = 0;  //new
          if(ii >= (t_lag + Calendar)/Tperiod) IFLAG[j][k][ii] = 0;  //new
                    if(ii < t_lag) {
                              ETc[j][k][ii] = 0.0f; //new
                              ER[ii] = 0.0f;
     }
     else if(ii >= t_lag){                              //new
               sumcycle[j][k] = (float)(sumcycle[j][k] + Tperiod);
               if(sumcycle[j][k] <= countday){
                         ETc[j][k][ii] = (float)( ((input_ETc[j][k][tt])/stage_day[tt])*Tperiod );
                              ER[ii] = (float)( ((Eff_rain[tt])/stage_day[tt])*Tperiod );
                         }
                         else{
                         if(stage_num[tt] != Calendar ){
                         ETc[j][k][ii] = (float)(odd2[j][tt]*(input_ETc[j][k][tt])/stage_day[tt] +
               (Tperiod - odd2[j][tt])*(input_ETc[j][k][tt+1])/stage_day[tt+1] );

                         ER[ii] = (float)(odd2[j][tt]*(Eff_rain[tt])/stage_day[tt] + (Tperiod -
               odd2[j][tt])*(Eff_rain[tt+1])/stage_day[tt+1] );
               }
      else {             ETc[j][k][ii] = (float)(odd2[j][tt]*(input_ETc[j][k][tt])/stage_day[tt] );
                         ER[ii] = (float)(odd2[j][tt]*(Eff_rain[tt])/stage_day[tt] );
     }
     tt++;
     countday = countday + stage_day[tt];
               }
     }
     // This is to adjust D_factor
               f(ETc[j][k][ii] > 5.0f){      //new for vary p factor
                         P[ii] = (float)(D_factr + 0.04*(5 - ETc[j][k][ii]));
                         if(P[ii] < 0.1) P[ii] = 0.1f;
                         if(P[ii] > 0.8) P[ii] = 0.8f;
               }
               else P[ii] = D_factr;
               //Set up smc in every soil strip layer as wilting point of every soil layer
                         FC_layer          = (float)(FC[j][k]*layerD*1000);
                         RAM_layer         = (float)((FC[j][k] - PWP[j][k])*layerD*1000*P[ii]);
                         WP_layer          = FC_layer - RAM_layer;

                         for(nsl = 0; nsl < MaxLyr; nsl++) {
                                   smc_layer[nsl][j][k][ii-1] = WP_layer;
                         }
               // new all for Kcb adjust and then cal of RD adjust
```

```
                if(ii < t_lag) RD[j][k][ii] = 0.0f;     //new
                else if(ii >= t_lag || smc_rz[j][k][ii-1] >= WP_RZ[j][k][ii-1]) {   //new
                if(sumcycle[j][k] < Lini) Kcb[ii] = Kcb_ini; // (0-24)
                if(sumcycle[j][k] >= Lini && sumcycle[j][k] < Lini+Ldev ){  //(25-49)
                        Kcb[ii] = Kcb_ini + ((sumcycle[j][k] - Lini)/Lini)*(Kcb_mid - Kcb_ini);
                }
        if(sumcycle[j][k] >= Lini+Ldev && sumcycle[j][k] < Lini+Ldev+Lmid) Kcb[ii] = Kcb_mid;
        if(sumcycle[j][k] >= Lini+Ldev+Lmid && sumcycle[j][k] < Calendar){ // (80-99)
          Kcb[ii] = Kcb_mid + (( sumcycle[j][k] - (Lini+Ldev+Lmid))/Llate)*(Kcb_end - Kcb_mid);
        }
        if(sumcycle[j][k] < Lini+Ldev){
          RD[j][k][ii] = MinRD + (MaxRD - MinRD)*(Kcb[ii] - Kcb_ini)/(Kcb_mid - Kcb_ini);
          if(RD[j][k][ii] > MaxRD) RD[j][k][ii] = MaxRD;
        }
        else{
          RD[j][k][ii] = MaxRD;
        }
        }
        else if(smc_rz[j][k][ii-1] < WP_RZ[j][k][ii-1]) {
        if(sumcycle[j][k] < Lini) Kcb[ii] = Ks[j][k][ii]*Kcb_ini;
        if(sumcycle[j][k] >= Lini && sumcycle[j][k] < Lini+Ldev ){
        Kcb[ii] = Ks[j][k][ii]*Kcb_ini + ((sumcycle[j][k] - Lini)/Lini)*(Ks[j][k][ii]*Kcb_mid -
        Ks[j][k][ii]*Kcb_ini);
        }
        if(sumcycle[j][k] >= Lini+Ldev && sumcycle[j][k] < Lini+Ldev+Lmid) Kcb[ii] =
        Ks[j][k][ii]*Kcb_mid;
        if(sumcycle[j][k] >= Lini+Ldev+Lmid && sumcycle[j][k] < Calendar){
                Kcb[ii] = Ks[j][k][ii]*Kcb_mid + (( sumcycle[j][k] -
                (Lini+Ldev+Lmid))/Llate)*(Ks[j][k][ii]*Kcb_end - Ks[j][k][ii]*Kcb_mid);
                }
        if(sumcycle[j][k] < Lini+Ldev){
                RD[j][k][ii] = MinRD + (MaxRD - MinRD)*(Kcb[ii] -
                Ks[j][k][ii]*Kcb_ini)/(Ks[j][k][ii]*Kcb_mid - Ks[j][k][ii]*Kcb_ini);
                if(RD[j][k][ii] > MaxRD) RD[j][k][ii] = MaxRD;
        }
        else RD[j][k][ii] = MaxRD;
}//end of Kcb and RD adjust
// Calculate supply start here with
sup_wrbd[j][k] = (float)(IrriEff*Scapa[j][k]*Tperiod*3600*24*1000)/(subLat_area[j][k]*10000);
        // For fixloss
        irrig[j][k][ii] = sup_wrbd[j][k] * IFLAG[j][k][ii];
                if(ii == t_lag) {
                        if(IFLAG[j][k][ii] == 1) {
                                irrig[j][k][ii] = (float)(sup_wrbd[j][k] - fixloss) * IFLAG[j][k][ii];
                        }
                }
                if(ii > t_lag) {
                        if(IFLAG[j][k][ii-1] == 0 && IFLAG[j][k][ii] == 1) {
                                irrig[j][k][ii] = (float)(sup_wrbd[j][k] - fixloss) * IFLAG[j][k][ii];
                        }
                }

                //new for sumup of fixloss
                if(ii > t_lag && ii < CYCLE+t_lag){
                        if(IFLAG[j][k][ii-1] == 1 && IFLAG[j][k][ii] == 0) {
                                countn = 1;
                                Nfix[j][k][ii] = (float)(fixloss * countn);
                        }
```

```
                    else Nfix[j][k][ii] = 0.0f;
                }
                else if(ii == CYCLE+t_lag-1){
                    if(IFLAG[j][k][ii-1] == 0 && IFLAG[j][k][ii] == 1){
                        countn = 1;
                        Nfix[j][k][ii] = (float)(fixloss * countn);
                    }
                    else Nfix[j][k][ii] = 0.0f;
                }
            }// end of ii loop preparation of all input data
            // Now start of Main loop in every cycle
            for(ii = 0; ii < cycle[i][j][k]; ii++) {// Main LOOP for ii                    new move
        // Start to strip soil layer in calculation here //new
                nlr_RZ_MAX              = (int)(MaxRD/layerD); //no. of layers at RD=Max
                NLR_RZ[j][k][ii]= (int)(RD[j][k][ii]/layerD); // no. of layers at a time period
                RD[j][k][ii]    = (float)(NLR_RZ[j][k][ii]*layerD); //adjust RD by layer
                FC_RZ[j][k][ii]  = FC[j][k]*RD[j][k][ii]*1000;
                PWP_RZ[j][k][ii]= PWP[j][k]*RD[j][k][ii]*1000;
                RAM_RZ[j][k][ii]= (FC[j][k] - PWP[j][k])*(RD[j][k][ii]*1000)*P[ii];
                WP_RZ[j][k][ii]  = FC_RZ[j][k][ii] - RAM_RZ[j][k][ii]; //new for varying p
                if(ii > t_lag){
                    if(NLR_RZ[j][k][ii] > NLR_RZ[j][k][ii-1]){
                        tsum[j][k][ii] = 0.0f;
                        for(nsl = NLR_RZ[j][k][ii-1]; nsl < NLR_RZ[j][k][ii]; nsl++){
                            tsum[j][k][ii] = tsum[j][k][ii] + smc_layer[nsl][j][k][ii-1];
                        }
                        smc_rz[j][k][ii] = smc_rz[j][k][ii-1] + tsum[j][k][ii];
                    }
                    else smc_rz[j][k][ii] = smc_rz[j][k][ii-1];
                }
                // This is to sum up smc for global check
            if(ii == t_lag) sum_up_smc[j][k][ii] = WP_RZ[j][k][ii];
            else sum_up_smc[j][k][ii] = smc_rz[j][k][ii];
            //now calculate supply in each t-period
            supply[j][k][ii] = irrig[j][k][ii]*IFLAG[j][k][ii];
            sumsupply[i] = sumsupply[i] + supply[j][k][ii] * subLat_area[j][k] / TotalAREA;
//Soil Water Balance Main Equation
if(ii==t_lag) {
            Ea[j][k][ii] = ETc[j][k][ii]; //because initial smc = WP_RZ
            smc_rz[j][k][ii]  = (float)(WP_RZ[j][k][ii] + supply[j][k][ii] - Ea[j][k][ii] + ER[ii]);
            if(smc_rz[j][k][ii] < WP_RZ[j][k][ii]){
            Ea[j][k][ii] = (smc_rz[j][k][ii] - PWP_RZ[j][k][ii])*(ETc[j][k][ii])/(WP_RZ[j][k][ii]-
PWP_RZ[j][k][ii]);
            smc_rz[j][k][ii] = (float)(WP_RZ[j][k][ii] + supply[j][k][ii] - Ea[j][k][ii] + ER[ii]);
}
}
else if(ii > t_lag){
            if(smc_rz[j][k][ii] >= WP_RZ[j][k][ii]) {
            Ea[j][k][ii] = ETc[j][k][ii];
            smc_rz[j][k][ii] = (float)(smc_rz[j][k][ii] + supply[j][k][ii] - Ea[j][k][ii] + ER[ii]);
}
else if(smc_rz[j][k][ii] < WP_RZ[j][k][ii]){
            Ea[j][k][ii] = (smc_rz[j][k][ii] - PWP_RZ[j][k][ii])*(ETc[j][k][ii])/(WP_RZ[j][k][ii]-
PWP_RZ[j][k][ii]);
            smc_rz[j][k][ii] = (float)(smc_rz[j][k][ii] + supply[j][k][ii] - Ea[j][k][ii] + ER[ii]);
            }
}
// end Soil Moisture Balance Main Equation and its adjustment.
```

```
        SMCline[j][k][ii] = smc_rz[j][k][ii]; //for ploting
// this is when SMC > FC
        infilt = 0.0f;
        if(smc_rz[j][k][ii] > FC_RZ[j][k][ii]){ // when smc_rz OVER FC_RZ > infil cal. here
        SMCline[j][k][ii] = FC_RZ[j][k][ii];
        infilt = smc_rz[j][k][ii] - FC_RZ[j][k][ii];
        infiltopup[j][k][ii] =  infilt;
        smc_rz[j][k][ii] =  FC_RZ[j][k][ii];
        for(nsl = NLR_RZ[j][k][ii]; nsl < nlr_RZ_MAX; nsl++) {
                smc_layer[nsl][j][k][ii] = smc_layer[nsl][j][k][ii-1] + infilt;
                if(smc_layer[nsl][j][k][ii] > FC_layer){
                        infilt = smc_layer[nsl][j][k][ii] - FC_layer;
                        smc_layer[nsl][j][k][ii] = FC_layer;
                }
                else {
                        infilt = 0.0;
                        break;
                }
        }
                Ovr_supl[j][k][ii] = infilt;
        }
        else{
                for(nsl = 0; nsl < nlr_RZ_MAX; nsl++) {
                        smc_layer[nsl][j][k][ii] = smc_layer[nsl][j][k][ii-1];
                }
                infiltopup[j][k][ii] = 0.0f;
                        Ovr_supl[j][k][ii] = 0.0f;
                }
                sumOversupl[i] = sumOversupl[i] + Ovr_supl[j][k][ii];
// This is for plotting smc line if oversupply
        if(Ovr_supl[j][k][ii] > 0.0f) SMCline[j][k][ii]  = FC_RZ[j][k][ii] + Ovr_supl[j][k][ii]; //new
// this is for calculation of Ks when smc_rz < WP_RZ
        if(smc_rz[j][k][ii] < WP_RZ[j][k][ii]) {
         Ks[j][k][ii] = (FC_RZ[j][k][ii] - smc_rz[j][k][ii])/(FC_RZ[j][k][ii]-WP_RZ[j][k][ii]);
        }
        else Ks[j][k][ii] = 1;
// this is a penalty function
        if(smc_rz[j][k][ii] < WP_RZ[j][k][ii]) {
                LWP[j][k][ii] = (float)(WP_RZ[j][k][ii] - smc_rz[j][k][ii]);
                Equity[j][k][ii] = (float)pow((R1*LWP[j][k][ii]),2);
//this is for plotting
                SMCline[j][k][ii] = WP_RZ[j][k][ii] - LWP[j][k][ii];
        }
        else {
                LWP[j][k][ii] = 0.0f;
                Equity[j][k][ii] = 0.0f;
        }
        sumLWP[i] = sumLWP[i] + LWP[j][k][ii];
        sumEquity[i] = sumEquity[i] + Equity[j][k][ii];
// This is to compute Global Balance Check
        sumsmc_rz[i][j][k]          = sumsmc_rz[i][j][k] + sum_up_smc[j][k][ii];
        sumX[i][j][k]               = sumX[i][j][k]     + supply[j][k][ii];
        sumETc[i][j][k]       = sumETc[i][j][k]    + ETc[j][k][ii];
        sumEa[i][j][k]          = sumEa[i][j][k]     + Ea[j][k][ii];
        sumSMCend[i][j][k]   = sumSMCend[i][j][k] + smc_rz[j][k][ii];
        sumO[i][j][k]       = sumO[i][j][k]     + Ovr_supl[j][k][ii];
        cum_stress[i][j][k]  = cum_stress[i][j][k]+ LWP[j][k][ii];
        sumNfix[i][j][k]     = sumNfix[i][j][k]   + Nfix[j][k][ii];
```

```
        suminfil[i][j][k]   = suminfil[i][j][k]  + infiltopup[j][k][ii];
        GlobalSMC[i][j][k]  = sumsmc_rz[i][j][k] + sumX[i][j][k] - sumEa[i][j][k] - suminfil[i][j][k]
- sumSMCend[i][j][k];
        sumIrr_day[i][j][k]  = sumIrr_day[i][j][k] + IFLAG[j][k][ii];
}// end of ii Main loop
}// end of  k
}// end of j main loop
//This is for capacity constraint in Branch and Main canal
        for(j = 0; j <NSCHEME; j++){//Bcapa constraint is computed here.
        for(ii=0; ii<CYCLE; ii++){  //new
        sumTertiaryFlow[j][ii] = 0.0f;
        for(k = 0; k <num_sublat[j]; k++){
                S_latGO[ii][k] = Scapa[j][k]*IFLAG[j][k][ii];
                sumTertiaryFlow[j][ii]  = sumTertiaryFlow[j][ii] + S_latGO[ii][k];
        }
        if(sumTertiaryFlow[j][ii] > Bcapa[j]) {
                capaConstr[j][ii] = (float)R2*(sumTertiaryFlow[j][ii] - Bcapa[j]);
        }
        else capaConstr[j][ii] = 0.0f;
                sumBcapaCon[i] =  sumBcapaCon[i] + (float)pow(capaConstr[j][ii],2);
        }// end of cycle ii
        } //end j
//new transfer IFLAG to IFLAG_B
        for(ii=0; ii<CYCLE; ii++) {
                for(j = 0; j <NSCHEME; j++) {  //Bcapa constraint is computed here
                   sumIFLAG[j][ii] = 0;
                   for(k = 0; k <num_sublat[j]; k++){
                          sumIFLAG[j][ii] = sumIFLAG[j][ii] + IFLAG[j][k][ii];
                }
                if(sumIFLAG[j][ii] > 0) IFLAG_B[j][ii] = 1;
                else IFLAG_B[j][ii] = 0;
                }
                }
//Main capa. constraint starts here
        for(ii=0; ii<CYCLE; ii++){
                sumSecondaryFlow[ii] = 0.0f;
        for(j = 0; j <NSCHEME; j++){
                B_GO[ii][j] = Bcapa[j]*IFLAG_B[j][ii];
                sumSecondaryFlow[ii] = sumSecondaryFlow[ii] + B_GO[ii][j];
        }
        }
        for(ii=0; ii<CYCLE; ii++){ //new
                if(sumSecondaryFlow[ii] > (main_capa*PercentFlow/100)){
                  MnCapaCon[ii] = R3*(sumSecondaryFlow[ii] - main_capa*PercentFlow/100);
                }
                else MnCapaCon[ii] = 0.0f;
                 sumMCapaCon[i] = sumMCapaCon[i] + (float)pow(MnCapaCon[ii],2);
                }

ftnss[i] = (float)(sumsupply[i] + sumEquity[i] + sumBcapaCon[i] + sumMCapaCon[i]);
sumftnss = sumftnss + ftnss[i];
if(bestfit  > ftnss[i]){
bestfit = ftnss[i];
psumLWP= sumLWP[i];
psumOVR = sumOversupl[i];
psumBcapa = sumBcapaCon[i];
psumMCapaCon = sumMCapaCon[i];
psumsupply = sumsupply[i];
```

```
psumEquity = sumEquity[i];

for(j = 0; j<NSCHEME; j++) {
        for(k = 0; k < num_sublat[j]; k++) {
        psumX[j][k] = sumX[i][j][k];
        psumD[j][k] = sumD[i][j][k];
        psumsmc_rz[j][k] = sumsmc_rz[i][j][k];
        pLWPscheme[j][k] = LWPscheme[i][j][k];
        psuminfil[j][k] = suminfil[i][j][k];
        psumEa[j][k] = sumEa[i][j][k];
        psumETc[j][k] = sumETc[i][j][k];
        psumSMCend[j][k] = sumSMCend[i][j][k];
        psumNfix[j][k] = sumNfix[i][j][k];
        psumO[j][k] = sumO[i][j][k];
        pstress[j][k] = cum_stress[i][j][k];
        pGlobalSMC[j][k] = GlobalSMC[i][j][k];
        psumIrr_day[j][k] = sumIrr_day[i][j][k];

for(ii=0; ii<CYCLE; ii++) {
        pFlag[j][k][ii] = IFLAG[j][k][ii];
        pETc[j][k][ii] = ETc[j][k][ii];
        pEa[j][k][ii] = Ea[j][k][ii];
        psupply[j][k][ii] = supply[j][k][ii];
        pdemand[j][k][ii] = demand[j][k][ii];
        psmc_rz[j][k][ii] = smc_rz[j][k][ii];
        pSMCline[j][k][ii] = SMCline[j][k][ii];
        pOvr_supl[j][k][ii] = Ovr_supl[j][k][ii];
        pLWP[j][k][ii] = LWP[j][k][ii];
        ptsum[j][k][ii] = tsum[j][k][ii];
}
}//end of k sublat
for(ii=0; ii<CYCLE; ii++) {
        pcapaConstr[j][ii] = capaConstr[j][ii];
        pFlag_B[j][ii] = IFLAG_B[j][ii];
}
}//end of j loop
}// end of if bestfit
}//end of i

if (gen == MaxGEN || DELTA_ftnss <= DELTA ) {
index = 999;
for(j = 0; j<NSCHEME; j++) {
fprintf(f1,"IFLAG_S ");
for(ii=0; ii<CYCLE; ii++) fprintf(f1,"%9d",pFlag_B[j][ii]); fprintf(f1,"\n");
for(k = 0; k < num_sublat[j]; k++) {
N_fixloss = (float)(psumNfix[j][k]/fixloss);
fprintf(f1,"Irrifrq%3.0f\n", N_fixloss);
fprintf(f1,"Secondary%2d Tertiary%2d Qflow result = %5.3f m3/s \n", j+1, k+1, Scapa[j][k]);
fprintf(f1,"IFLAG ");
        for(ii=0; ii<CYCLE; ii++) fprintf(f1,"%9d",pFlag[j][k][ii]);
        fprintf(f1, " Irri_day %9.3f sumFixloss %9.3f\n", psumIrr_day[j][k], psumNfix[j][k]);
        fprintf(f1,"FC ");
for(ii=0; ii<CYCLE; ii++) fprintf(f1,"%9.3f",FC_RZ[j][k][ii]); fprintf(f1,"\n");

fprintf(f1,"Supply ");
      for(ii=0; ii<CYCLE; ii++) fprintf(f1,"%9.3f",psupply[j][k][ii]); fprintf(f1,"\n");
fprintf(f1,"ETc ");
      for(ii=0; ii<CYCLE; ii++) fprintf(f1,"%9.3f",pETc[j][k][ii]); fprintf(f1,"\n");
```

```
fprintf(f1,"Ea      ");
        for(ii=0; ii<CYCLE; ii++)  fprintf(f1,"%9.3f",pEa[j][k][ii]); fprintf(f1,"\n");
fprintf(f1,"smc_rz  ");
        for(ii=0; ii<CYCLE; ii++)  fprintf(f1,"%9.3f",psmc_rz[j][k][ii]);fprintf(f1,"\n");
fprintf(f1,"SMC     ");
        for(ii=0; ii<CYCLE; ii++)  fprintf(f1,"%9.3f",pSMCline[j][k][ii]);fprintf(f1,"\n");
fprintf(f1,"WP      ");
        for(ii=0; ii<CYCLE; ii++)  fprintf(f1,"%9.3f",WP_RZ[j][k][ii]); fprintf(f1,"\n");
fprintf(f1,"PWP     ");
        for(ii=0; ii<CYCLE; ii++)  fprintf(f1,"%9.3f",PWP_RZ[j][k][ii]); fprintf(f1,"\n");
fprintf(f1,"BelowWP ");
        for(ii=0; ii<CYCLE; ii++)  fprintf(f1,"%9.3f",pLWP[j][k][ii]);
fprintf(f1, " cum_stress %9.3f\n", pstress[j][k]);
fprintf(f1,"Global  %9.3f\n", pGlobalSMC[j][k]);
fprintf(f1,"SMCstr  %9.3f\n", psumsmc_rz[j][k]);
fprintf(f1,"SMCend  %9.3f\n", psumSMCend[j][k]);
fprintf(f1,"Irriga  %9.3f\n", psumX[j][k]);
fprintf(f1,"ETc     %9.3f\n", psumETc[j][k]);
fprintf(f1,"Ea_     %9.3f\n", psumEa[j][k]);
fprintf(f1,"infilt  %9.3f\n", psuminfil[j][k]);
fprintf(f1,"Drainage%9.3f\n", psumO[j][k]);
fprintf(f1,"stress  %9.3f\n", pstress[j][k]);
fprintf(f1,"lwrPWP  %9.3f\n", psumDie[j][k]);
} // end of if k
} //end of if j
fprintf(f1,"%4.2f  %4.3f", PXOVER, NofMu);
fprintf(f1," %4d  %7.3f  %7.3f  %7.3f %2d  %7.3f %2d  %7.3f  %2d  %7.3f\n",
 gen, ftnss[POPSIZE], bestfit, psumsupply, R1, psumEquity,  R2, psumBcapa, R3, psumMCapaCon);
 }// end of if loop
        avgftnss = sumftnss/POPSIZE;
        minftnss  = ftnss[0];
        for(i=0; i<POPSIZE; i++) {
                if(ftnss[i]  < minftnss)   minftnss  = ftnss[i];
                if(ftnss[i]  < ftnss[POPSIZE]) ftnss[POPSIZE] = ftnss[i];
        }
}// end of objective
/*******************************/
void Tournament() {
        int ii,jj,j,k,mem;
        for(mem=0; mem < POPSIZE; mem++){
                ii = RandInt(0,POPSIZE-1);
                jj = RandInt(0,POPSIZE-1);
                if(ftnss[ii] <= ftnss[jj]){
                        for(j=0; j<NSCHEME; j++){
                                for(k=0; k<LENGTH; k++){
                                        mated_pop[mem][j][k] = initial_pop[ii][j][k];
                                }
                        }
                }
    if(ftnss[ii] > ftnss[jj]){
                        for(j=0; j<NSCHEME; j++){
                                for(k=0; k<LENGTH; k++){
                                        mated_pop[mem][j][k] = initial_pop[jj][j][k];
                                }
                        }
                }
        }
}// end of tournament
```

```
/*********************************/
// Routine to crossover population
void XOVER(int i1, int i2) {
        int j, k, kk;
        kk = RandInt(1,LENGTH-1);
for(j = 0; j<NSCHEME; j++){
        for(k =kk; k<LENGTH; k++) {
        SwapRealValues(&mated_pop[i1][j][k],&mated_pop[i2][j][k]);
        }
}
}
/*********************************/
void UNIXOVER(int i1, int i2) {
int j, k, kk;
 for(j= 0; j<NSCHEME; j++){
        for(k = 0; k<LENGTH; k++) {
                kk= rand()&01;
                if(kk==1) SwapRealValues(&mated_pop[i1][j][k],&mated_pop[i2][j][k]);
        }
 }
}
/*********************************/
// Routine to crossover population, 2-site
void TPXOVER(int i1, int i2) {
        int j,k,kk1,kk2;
        kk1 = RandInt(1,LENGTH-1);
        kk2 = RandInt(1,LENGTH-1); // this is for 2-site XOVER
        if (kk1>kk2) SwapIntValues(&kk1,&kk2); // this is to swap 2-site if the later is less
 for(j= 0; j<NSCHEME; j++){
        for(k=kk1;k<kk2;k++) {
                SwapRealValues(&mated_pop[i1][j][k],&mated_pop[i2][j][k]);
  }
 }
}
/*************************************/
void SelectMate() {
int  j,k,mem, num_select,one;
float Prob_X;
  num_select = 0;
        for (mem = 0; mem< POPSIZE ; mem++) {
                Prob_X = rand()%1000/1000.0f;
                if (Prob_X < PXOVER){
                        ++num_select;
                        if(num_select%2 == 0){
                                XOVER(one,mem);
                        }
                        else one = mem;
                }
        }
}
/*****************************/
void UniSelectMate() {
int  j, k, mem, num_select,one;
  float Prob_X;
  num_select = 0;
        for (mem = 0; mem< POPSIZE ; mem++) {
                Prob_X = rand()%1000/1000.0f;
                if (Prob_X < PXOVER){
```

```
                        ++num_select;
                        if(num_select%2 == 0){
                                UNIXOVER(one,mem);
                        }
                        else one = mem;
                }
        }
}
/*****************************************/
// this is to random probability of mutation
void mutatepop() {
 int i,j,k;
 double r;
        for (i=0;i<POPSIZE;i++) {
                for(j=0; j<NSCHEME; j++) {
                        for(k= 0; k<LENGTH; k++){
                        r = rand()%1000/1000.0;
                        if(r<Pmutate){
                                if(mated_pop[i][j][k] == 0)  mated_pop[i][j][k] = 1;
                                else mated_pop[i][j][k] = 0;
                        }
                        }
                }
                for(j=0;j<NSCHEME;j++)  {
                        for(k = 0; k<LENGTH; k++) initial_pop[i][j][k] = mated_pop[i][j][k];
                }
        }//end of i POPSIZE
}
/*************************************/
// Routine to print error message
void nrerror(char *messg) {
        puts(messg) ;
   exit(1) ;
}
/*************************************/
// Routine to swop value
void SwapIntValues(int *x, int *y) {
        int temp;
        temp = *x;
        *x=*y;
        *y=temp;
}
/*************************************/
// Routine to swop real value
void SwapRealValues(float *xx, float*yy) {
        float temp;
        temp = *xx;
        *xx = *yy;
        *yy = temp;
}
/*************************************/
```

```
//GA46W.c          Program to formulate GAs for water scheduling problem
//                 WARABANDI approach (A more complex system, chapter 7)
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <process.h>
#include <time.h>

#define Tperiod      1.0     // input time period to balance water (0.5, 1.0...day)
#define POPSIZE      100     //input population size
#define INTERVAL     300     // input interval of generation to recheck of improvement fitness
#define DELTA  0.0001 // input minimum improvement of fitness after INTERVAL generation
#define  fixloss     1.5     // fixloss of each teriary canal
#define PercentFlow  100     // input percent of main canal flow of full capacity
#define NUMCROP      1       // input number of crop in this system
#define MAXSUBLAT    4       // input maximum tertiaries in a secondary
#define NSCHEME      4       // input number of scheme length
#define SumSubLat    16      // input sum of tertiaries in the system
#define Max_wrbd     7       // input maximum warabandi
#define Min_wrbd     1       // input minimum warabandi
#define MaxLag       15      // input maximum staggering days
#define MinLag       0       // input minimum staggering days
#define Calendar     100     // input total growth period
#define STAGE        100     // stage of growth define related to crop requirement data file
#define CYCLE        115     // input total time step = ceil(Calendar/Tperiod)+MaxLag
#define LENGTH       3       // input chromosome length for each tertiary
#define inc          Tperiod // increment of 1.0 day to mutate wara and shutdown (day)
#define IrriEff      0.837
#define layerD       0.005   // in meter        new
#define tDepth       1.0     // in meter        new
#define MaxLyr       200     // new

int      RandInt (int, int);
float    RandReal (float,float);
void     Read_inputfiles();
void     SwapIntValues(int *, int *);
void   SwapRealValues(float *, float *);
void     nrerror(char *messg) ;
void     get_f(char name[]) ;
void     initial();
void     objective();
void     SelectMate();
void     UniSelectMate();
void     mutatepop();
void   Tournament();

int      MaxGEN, R1, R2, R3;
float    t_lag;
float    NofMu, PXOVER;

int      gen, i ,tt, index, stage_day[STAGE], num_sublat[NSCHEME];
int      g, stage_num[STAGE];
float    mated[POPSIZE],bestfit, Pmutate, ftnss[POPSIZE+1], DELTA_ftnss, OLD_minftnss;
float    initial_pop[POPSIZE][NSCHEME][MAXSUBLAT][LENGTH],
         mated_pop[POPSIZE][NSCHEME][MAXSUBLAT][LENGTH];
```

```
float    MinRD, MaxRD, Lini, Ldev, Lmid, Llate, Kcb_ini, Kcb_mid, Kcb_end, Kcb[CYCLE];
float    RootD[STAGE], D_factr, FC[NSCHEME][MAXSUBLAT],
         PWP[NSCHEME][MAXSUBLAT];
float    Bcapa[NSCHEME],loss_fct[NSCHEME], input_ETc[MaxLag][STAGE], main_capa;
float    Scapa[NSCHEME][MAXSUBLAT], scheme_area[NSCHEME], Eff_rain[STAGE];
float    subLat_area[NSCHEME][MAXSUBLAT], TotalAREA;
float    strt[NSCHEME][MAXSUBLAT], pstrt[NSCHEME][MAXSUBLAT];
float    wara[NSCHEME][MAXSUBLAT], pwara[NSCHEME][MAXSUBLAT];
float    shut[NSCHEME][MAXSUBLAT], pshut[NSCHEME][MAXSUBLAT];

FILE *f1,*f2,*f3 ;

//MAIN PROGRAM
void main (void) {
  clock_t start, finish;
  float  duration, TotalTime ;

  TotalTime = 0.0f;
  if(!(f1 = fopen("ga46W.out","w"))) { //ga46W.out for warabandi output
              printf("can't open file ga46w.out\n");
              exit(1);
       }
Read_inputfiles();
srand(1);
start = clock();

MaxGEN=      2500;
PXOVER=      0.9f;
NofMu  =     0.05f;
R1     =     1;
R2     =     10000;
R3     =     12500;
//     R1    =  3;
//do{
//     R2           =     2500;
//do{
//     R3           =     15000;
//do{
//     NofMu  =     0.05f;
//do{
//     PXOVER       =     0.5f;
//do{
  initial();
  index = 0;
       OLD_minftnss   = 10000.0f;
       DELTA_ftnss    = 10000.0f;
       bestfit        = 100000.0f;
       ftnss[POPSIZE] = 1000000.0f;
 for(gen = 0; gen < MaxGEN; gen++) {
       objective(); if(index == 999) break;
       Tournament();
//     SelectMate();
       UniSelectMate();
  mutatepop();
  if (!((gen+1)%INTERVAL)){
    DELTA_ftnss = (float)fabs((double)((OLD_minftnss - ftnss[POPSIZE])/ftnss[POPSIZE]));
    OLD_minftnss = ftnss[POPSIZE];
  }
```

```
  } //End of Generation Loop
//  PXOVER = (float) (PXOVER + 0.05f);
//}while (PXOVER <= 1.01);
//NofMu = (float)(NofMu + 0.05f);
//}while (NofMu <= 0.11);
//R3 = R3 + 2500;
//}while (R3 <= 17501);
//R2 = R2 + 2500;
//}while (R2 <= 12501);
//R1 = R1 + 2;
//}while (R1 <= 8);
  finish = clock();
  duration = (float)(finish - start) / CLOCKS_PER_SEC;
  TotalTime = (float)(TotalTime + duration);
  printf(" Time = %5.2f minutes\n", duration/60);
  fprintf(f1,"Time = %5.2f minutes \n\n", duration/60);
  (void) fclose (f1);
  printf("End of run  ENJOY!!!! ");
} //MAIN PROGRAM END
/***************************/
void Read_inputfiles() {
   int   ii, jj, kk, cc, num_schemes, scheme_num[NSCHEME];
         int   scheme_num1[NSCHEME], scheme_num2[NSCHEME], scheme_num4[NSCHEME],
scheme_num5[NSCHEME];
         int   num_crop, crop_nmbr2,crop_period[NUMCROP];
         int   sublat[MAXSUBLAT], Nstage, crop_nmbr;
         float crop_area[NUMCROP][NSCHEME];
         char  title1[80],title2[80],title3[80],title6[80], title7[80];
         char  title8[80], title9[80], title10[80], title11[80];
         char  schedulefile[20];
         if(!(f2 = fopen("Proj46W.lst","r"))) {     //for input46A.dat
     nrerror(" Can not open project file") ;
   }
   // start reading file here
         fgets(schedulefile, 20, f2) ;
         for(ii = 0; ii < 20; ii++) {
                  if(schedulefile[ii] == '\n') schedulefile[ii] = '\0' ;
         }
         (void) fclose(f2) ;
   // now read water scheduling data
         if(!(f2 = fopen(schedulefile,"r"))) {
     nrerror(" Unable to open schedulefile") ;
   }
// Canal data file start here **********************
   fgets(title1, 80, f2);        // Canal data title
   fscanf(f2,"%d",&num_schemes);  // scan number of schemes here
   for(jj=0;jj<num_schemes;jj++) fscanf(f2,"%d ", &scheme_num[jj]);
         for(jj = 0; jj < NSCHEME; jj++) {
                  fscanf(f2,"%d", &num_sublat[jj]);
         }                                                                    //4     4        4 4
         fscanf(f2,"\n");
         fscanf(f2,"%d",&num_crop);  // number of crop (how many crops in this system
         for(jj = 0; jj < NSCHEME; jj++) {
    fscanf(f2,"%d",&scheme_num1[jj]);
           for(kk = 0; kk < num_sublat[jj]; kk++){
      fscanf(f2,"%d", &sublat[kk]);
                  for (cc = 0; cc < num_crop; cc++){
                           fscanf(f2,"%f",&crop_area[cc][jj]);
```

```
                }
            }
        }
        fscanf(f2,"\n");
TotalAREA = 0.0f;
for(jj = 0; jj < NSCHEME; jj++) {
        scheme_area[jj] = 0.0f;
          for(kk = 0; kk<num_sublat[jj]; kk++){
                subLat_area[jj][kk] = 0.0f;
                        for (cc = 0; cc < num_crop; cc++){
                                subLat_area[jj][kk] = subLat_area[jj][kk] + crop_area[cc][jj];
                        }
                        scheme_area[jj] = scheme_area[jj] + subLat_area[jj][kk];
                }
        TotalAREA = TotalAREA + scheme_area[jj];
        }
  fgets(title2, 80, f2);        // Main & lateral canals title
        fscanf(f2,"%f ", &main_capa) ; // scan main canal supply

  // scan branch canal supply
        for(jj = 0; jj < num_schemes; jj++) fscanf(f2,"%f ", &Bcapa[jj]) ;
        fscanf(f2,"\n");
  //scan sub-lateral capa
        fgets(title3, 80, f2);
        for(jj = 0; jj < NSCHEME; jj++) {
                fscanf(f2,"%d ", &scheme_num2[jj]);
                for(kk = 0; kk < num_sublat[jj]; kk++) {
                        fscanf(f2,"%f ", &Scapa[jj][kk]);
                }
        }
  fscanf(f2,"\n");
// Soil data file start here ******************
        fgets(title6, 80, f2);        // field capacity title
        for(jj = 0; jj < num_schemes; jj++) {
                fscanf(f2,"%d ", &scheme_num4[jj]);
        for(kk = 0; kk < MAXSUBLAT; kk++){
                  fscanf(f2,"%f ", &FC[jj][kk]) ;    // scan FC
                }
        }
  fscanf(f2,"\n");
  fgets(title7, 80, f2);        // permanent wilting point title
        for(jj = 0; jj < num_schemes; jj++) {
                fscanf(f2,"%d ", &scheme_num5[jj]);
        for(kk = 0; kk < MAXSUBLAT; kk++){
                  fscanf(f2,"%f ", &PWP[jj][kk]) ;   // scan PWP
                }
        }
  fscanf(f2,"\n");
        // Crop Data File start here  ******************
  fgets(title8, 80, f2);                                        // Detail crop datas
        fscanf(f2,"%3d",&crop_nmbr);
        for (cc = 0; cc < crop_nmbr; cc++){
                fscanf(f2,"%3d",&crop_period[cc]);//Days of Max. crop calender (100)
        }
        fscanf(f2,"%3d",&crop_nmbr2);    // crop number (1)
        fscanf(f2,"%3d ",&Nstage);                  /************************/
        for(tt = 0; tt < Nstage; tt++){
                fscanf(f2,"%3d",&stage_num[tt]);            // total stage number = 1,2,3....100
```

```
        }
        fscanf(f2,"\n");
        for(tt = 0; tt < Nstage; tt++){
                fscanf(f2,"%2d",&stage_day[tt]);          // number of days in each stage (1)
        }
        fscanf(f2,"\n");
for(kk = 0; kk < MaxLag; kk++){
                for(tt = 0; tt < Nstage; tt++){
                        fscanf(f2,"%f", &input_ETc[kk][tt]);        //cnvrs_ETc[jj][kk][tt]
                }
               }
  fscanf(f2,"\n");
        for(tt = 0; tt <Nstage; tt++){                  // Effective rainfall
          fscanf(f2,"%f", &Eff_rain[tt]) ;
        }
  fscanf(f2,"\n");
  fgets(title9,80,f2);   // Length stage title
        fscanf(f2,"%f %f %f %f",&Lini, &Ldev, &Lmid, &Llate);
  fscanf(f2,"\n");
        fgets(title10,80,f2);   // Kcb title
        fscanf(f2,"%f %f %f",&Kcb_ini, &Kcb_mid, &Kcb_end);
  fscanf(f2,"\n");
  fgets(title11, 80, f2); // rooting depth title
        fscanf(f2,"%f %f", &MinRD, &MaxRD);
        fscanf(f2,"%f",&D_factr);          // Defraction factors
  (void)fclose(f2);
}//end Readinput file
/*********/
float RandReal (float lower,float upper) {
        float val;
        float nb = ( upper - lower );
        val =  (float)(rand()%1000/1000.0*nb + lower);
        return ((float)val);
}
/*********/
int RandInt (int lower,int upper) {
        int val;
        int nb = (upper - lower );
        val =  (rand()%nb + lower);
        return val;
}
/*********/
void initial() {
  int i, j, k;
  float nd = RAND_MAX;

  for(i = 0; i<POPSIZE; i++) {
          for(j = 0; j< NSCHEME; j++){
                  for(k = 0; k< MAXSUBLAT; k++){
                          initial_pop[i][j][k][0] = (float)(MinLag  + inc*(float)floor((1+((MaxLag-
                          1) - MinLag)/inc)*(float)rand()/(nd+1.00)));

                          initial_pop[i][j][k][1] = (float)(Min_wrbd +
                          inc*(float)floor((1+(Max_wrbd -
                          Min_wrbd)/inc)*(float)rand()/(nd+1.00)));
```

```
                        initial_pop[i][j][k][2] = (float)(Min_wrbd +
                        inc*(float)floor((1+(Max_wrbd -
                        Min_wrbd)/inc)*(float)rand()/(nd+1.00)));
                }
        }
  }
/************************/
void objective() {
int     i, j, k, ii, cycle[POPSIZE][NSCHEME][MAXSUBLAT], countday;
int  IFLAG[NSCHEME][MAXSUBLAT][CYCLE], pFlag[NSCHEME][MAXSUBLAT][CYCLE];
int     IFLAG_B[NSCHEME][CYCLE], pFlag_B[NSCHEME][CYCLE];
float   sumftnss, minftnss, avgftnss;
float   odds[POPSIZE][NSCHEME], odd2[NSCHEME][STAGE],
        sumcycle[NSCHEME][MAXSUBLAT];
int     cvrs;
float   sumii, rota[NSCHEME][MAXSUBLAT];
float   Ovr_supl[NSCHEME][MAXSUBLAT][CYCLE],
pOvr_supl[NSCHEME][MAXSUBLAT][CYCLE];
        float    LWP[NSCHEME][MAXSUBLAT][CYCLE],
        pLWP[NSCHEME][MAXSUBLAT][CYCLE];
float   S_latGO[CYCLE][MAXSUBLAT];
float   pETc[NSCHEME][MAXSUBLAT][CYCLE];
float   ETc[NSCHEME][MAXSUBLAT][CYCLE], sup_wrbd[NSCHEME][MAXSUBLAT];
float   supply[NSCHEME][MAXSUBLAT][CYCLE],
        psupply[NSCHEME][MAXSUBLAT][CYCLE];
float   demand[NSCHEME][MAXSUBLAT][CYCLE],
        pdemand[NSCHEME][MAXSUBLAT][CYCLE];
float   psumOVR, psumLWP, psumBcapa, RD[NSCHEME][MAXSUBLAT][CYCLE];
float   sumBcapaCon[POPSIZE], sumLWP[POPSIZE];
float   capaConstr[NSCHEME][CYCLE], pcapaConstr[NSCHEME][CYCLE];
float  GlobalSMC[POPSIZE][NSCHEME][MAXSUBLAT],
        pGlobalSMC[NSCHEME][MAXSUBLAT];
float   sumX[POPSIZE][NSCHEME][MAXSUBLAT],
        sumD[POPSIZE][NSCHEME][MAXSUBLAT];
float   psumD[NSCHEME][MAXSUBLAT], psumX[NSCHEME][MAXSUBLAT];
        float    sumsmc_rz[POPSIZE][NSCHEME][MAXSUBLAT],
        psumsmc_rz[NSCHEME][MAXSUBLAT];
float   smc_rz[NSCHEME][MAXSUBLAT][CYCLE],
        psmc_rz[NSCHEME][MAXSUBLAT][CYCLE];
float   sumO[POPSIZE][NSCHEME][MAXSUBLAT], sumGlobal[POPSIZE],
        sumsupply[POPSIZE];
float   MnCapaCon[CYCLE], sumMCapaCon[POPSIZE], psumMCapaCon;
float   sumTertiaryFlow[NSCHEME][CYCLE], sumSecondaryFlow[CYCLE];
float   SMCline[NSCHEME][MAXSUBLAT][CYCLE],
        pSMCline[NSCHEME][MAXSUBLAT][CYCLE];
float   sumEquity[POPSIZE], psumsupply, psumEquity,
        Equity[NSCHEME][MAXSUBLAT][CYCLE];
float   sumOversupl[POPSIZE];
float   P[CYCLE], Ks[NSCHEME][MAXSUBLAT][CYCLE];
int     NLR_RZ[NSCHEME][MAXSUBLAT][CYCLE], nlr_RZ_MAX, nsl, countn;
float   FC_layer, RAM_layer, WP_layer, RAM_RZ[NSCHEME][MAXSUBLAT][CYCLE];
float   FC_RZ[NSCHEME][MAXSUBLAT][CYCLE],
        PWP_RZ[NSCHEME][MAXSUBLAT][CYCLE],
        WP_RZ[NSCHEME][MAXSUBLAT][CYCLE];
float   smc_layer[200][NSCHEME][MAXSUBLAT][CYCLE],
        tsum[NSCHEME][MAXSUBLAT][CYCLE];
float   pFC_RZ[NSCHEME][MAXSUBLAT][CYCLE],
        prnWP_RZ[NSCHEME][MAXSUBLAT][CYCLE];
```

```
float     prnPWP_RZ[NSCHEME][MAXSUBLAT][CYCLE];
float  Ea[NSCHEME][MAXSUBLAT][CYCLE], infilt ; //new
float  pstress[NSCHEME][MAXSUBLAT], cum_stress[POPSIZE][NSCHEME][MAXSUBLAT];
float  ER[CYCLE], irrig[NSCHEME][MAXSUBLAT][CYCLE],
                Nfix[NSCHEME][MAXSUBLAT][CYCLE];
float     sum_up_smc[NSCHEME][MAXSUBLAT][CYCLE];
float     infiltopup[NSCHEME][MAXSUBLAT][CYCLE];
float     sumNfix[POPSIZE][NSCHEME][MAXSUBLAT],
          sumEa[POPSIZE][NSCHEME][MAXSUBLAT];
float     suminfil[POPSIZE][NSCHEME][MAXSUBLAT],
          sumSMCend[POPSIZE][NSCHEME][MAXSUBLAT];
float     sumIrr_day[POPSIZE][NSCHEME][MAXSUBLAT],
          psuminfil[NSCHEME][MAXSUBLAT];
float     psumEa[NSCHEME][MAXSUBLAT],psumSMCend[NSCHEME][MAXSUBLAT];
float     psumNfix[NSCHEME][MAXSUBLAT], psumO[NSCHEME][MAXSUBLAT];
float     psumIrr_day[NSCHEME][MAXSUBLAT], pEa[NSCHEME][MAXSUBLAT][CYCLE] ;
float     ptsum[NSCHEME][MAXSUBLAT][CYCLE], N_fixloss, B_GO[CYCLE][NSCHEME];
float     sumIFLAG[NSCHEME][CYCLE],
sumETc[POPSIZE][NSCHEME][MAXSUBLAT],psumETc[NSCHEME][MAXSUBLAT];
float     cnvrs_ETc[NSCHEME][MAXSUBLAT][STAGE];
int    pcycle[NSCHEME][MAXSUBLAT];

          sumftnss = 0.0f;
for(i=0;i<POPSIZE;i++)  {
          ftnss[i]            = 0.0f;
          sumLWP[i]                   = 0.0f;
          sumOversupl[i]  = 0.0f;
          sumBcapaCon[i] = 0.0f;
          sumMCapaCon[i]          = 0.0f;
          sumsupply[i]      = 0.0f;
          sumGlobal[i]      = 0.0f;
          sumEquity[i]      = 0.0f;
// now transfer initial pop
          for(j = 0; j < NSCHEME; j++){
          for(k = 0; k < num_sublat[j]; k++) {
                              strt[j][k] = initial_pop[i][j][k][0];
                              wara[j][k] = initial_pop[i][j][k][1];
                              shut[j][k] = initial_pop[i][j][k][2];
                              t_lag = strt[j][k];
                              cycle[i][j][k] = (int)((Calendar/Tperiod)+t_lag);
                    // compute supply from this point
sup_wrbd[j][k] = (float)(IrriEff*Scapa[j][k]*Tperiod*3600*24*1000)/(subLat_area[j][k]*10000);
}
          odds[i][j] = (float)(Calendar - (Tperiod*(int)floor(Calendar/Tperiod)) );
          Pmutate    = (float)(NofMu/(LENGTH*SumSubLat));
// LENGTH = length of a tertiary, a chromosome length is then equal to LENGTH*SumSubLat
}
// now transfer cnvrs_ETc by strt[j][k]
          for(j = 0; j <NSCHEME; j++) {
          for(k = 0; k < num_sublat[j]; k++){
                    cvrs = (int)(strt[j][k]);
                    for(tt = 0; tt < STAGE; tt++){
                              cnvrs_ETc[j][k][tt] = input_ETc[cvrs][tt];
                    }
                   }
          }
// every stage of month it has odd2 that occours when devide stageday with Tperiod
```

```
        for(j = 0; j < NSCHEME; j++){ // LOOP J2
    for(tt = 0; tt < STAGE; tt++){
        if(tt == 0) {
                odd2[j][tt] = (float)(stage_day[tt] - ((int)floor(stage_day[tt]/Tperiod))*Tperiod );
        }
        else{
        odd2[j][tt] = (float)(odd2[j][tt-1] + stage_day[tt] - ((int)floor((odd2[j][tt-1] + stage_day[tt]) /
        Tperiod))*Tperiod );
        }
        }
        } // end J2
// new transfer array is here
        for(j = 0; j < NSCHEME; j++){     // new all loop
                for(k = 0; k < num_sublat[j]; k++){
                rota[j][k] = wara[j][k] + shut[j][k];
                t_lag  = strt[j][k];
                sumii    =        0.0f;
                for(ii    = 0; ii< CYCLE; ii++){  // new approved
                        sumii++;
                        IFLAG[j][k][ii] = 0;
                        if(sumii >  t_lag && sumii <= wara[j][k]+t_lag) IFLAG[j][k][ii] = 1;
                                if(sumii >  wara[j][k]+t_lag) {
                                        if(sumii <= rota[j][k]+t_lag) IFLAG[j][k][ii] = 0;
                                        else {
                                                IFLAG[j][k][ii] = 1;  //new
                                                sumii = t_lag+ 1.0f;
                                        }
                                }
                        if(ii >= t_lag+Calendar) IFLAG[j][k][ii] = 0; //new!
                        }
                }
        }       // end new transfer array
/**********      Transfer Array end          ****/
        for(j = 0; j < NSCHEME; j++) { //MAIN SCHEME LOOP for every branch OPEN HERE
        for(k = 0; k < num_sublat[j]; k++) { // MAIN LOOP for k is HERE
                sumX[i][j][k]     = 0.0f;
                sumEa[i][j][k]    = 0.0f;
                sumETc[i][j][k] = 0.0f;
                suminfil[i][j][k]  = 0.0f;
                sumDie[i][j][k]   = 0.0f;
                sumO[i][j][k]     = 0.0f;
                sumsmc_rz[i][j][k]= 0.0f;
                cum_stress[i][j][k] = 0.0f;
                sumIrr_day[i][j][k] = 0.0f;
                sumSMCend[i][j][k]  = 0.0f;
                sumNfix[i][j][k]  = 0.0f;
    tt = 0;
    sumcycle[j][k] = 0;
        countday =  stage_day[0];
        t_lag = strt[j][k];
        for(ii = 0; ii< cycle[i][j][k]; ii++){   // MAIN LOOP CYCLE is HERE
                        if(ii <  t_lag) IFLAG[j][k][ii] = 0;
                        if(ii >= t_lag+Calendar) IFLAG[j][k][ii] = 0;
                if(ii < t_lag) {
                        ETc[j][k][ii] = 0.0f; //new
                        ER[ii] = 0.0f;
                }
                else if(ii >= t_lag){
```

```
                        sumcycle[j][k] = (float)(sumcycle[j][k] + Tperiod);
                        if(sumcycle[j][k] <= countday){
                         ETc[j][k][ii] = (float)( (cnvrs_ETc[j][k][tt]/stage_day[tt])*Tperiod );
                         ER[ii] = (float)( (Eff_rain[tt]/stage_day[tt])*Tperiod );
                        }
                        else{
                        if(stage_num[tt] != STAGE ){
                        ETc[j][k][ii] = (float)(odd2[j][tt]*(cnvrs_ETc[j][k][tt])/stage_day[tt] +
                        (Tperiod - odd2[j][tt])*(cnvrs_ETc[j][k][tt+1])/stage_day[tt+1] );

                        ER[ii] = (float)(odd2[j][tt]*(Eff_rain[tt])/stage_day[tt] + (Tperiod -
                        odd2[j][tt])*(Eff_rain[tt+1])/stage_day[tt+1] );
                        }
                        else {
                        ETc[j][k][ii] = (float)(odd2[j][tt]*(cnvrs_ETc[j][k][tt])/stage_day[tt] );
                        ER[ii] = (float)(odd2[j][tt]*(Eff_rain[tt])/stage_day[tt] );
                        }
                        tt++;
                        countday = countday + stage_day[tt];
                        }
                }
// This is to adjust D_factor
                if(ETc[j][k][ii] > 5.0f){     //new for vary p factor
                         P[ii] = (float)(D_factr + 0.04*(5 - ETc[j][k][ii]));
                         if(P[ii] < 0.1) P[ii] = 0.1f;
                         if(P[ii] > 0.8) P[ii] = 0.8f;
                }
                else P[ii] = D_factr;
//Set up smc in every soil strip layer as wilting point of every soil layer
                        FC_layer        = (float)(FC[j][k]*layerD*1000);
                        RAM_layer       = (float)((FC[j][k] - PWP[j][k])*layerD*1000*P[ii]);
                        WP_layer        = FC_layer - RAM_layer;
                        for(nsl = 0; nsl < MaxLyr; nsl++) {
                                smc_layer[nsl][j][k][ii-1] = WP_layer;
                        }
// new all for Kcb adjust and then cal of RD adjust
if(ii < t_lag) RD[j][k][ii] = 0.0f;
else if(ii >= t_lag || smc_rz[j][k][ii-1] >= WP_RZ[j][k][ii-1]) {
if(sumcycle[j][k] < Lini) Kcb[ii] = Kcb_ini; // (0-24)
if(sumcycle[j][k] >= Lini && sumcycle[j][k] < Lini+Ldev ){  //(25-49)
   Kcb[ii] = Kcb_ini + ((sumcycle[j][k] - Lini)/Lini)*(Kcb_mid - Kcb_ini);
}
if(sumcycle[j][k] >= Lini+Ldev && sumcycle[j][k] < Lini+Ldev+Lmid) Kcb[ii] = Kcb_mid;
if(sumcycle[j][k] >= Lini+Ldev+Lmid && sumcycle[j][k] < Calendar){ // (80-99)
   Kcb[ii] = Kcb_mid + (( sumcycle[j][k] - (Lini+Ldev+Lmid))/Llate)*(Kcb_end - Kcb_mid);
}
if(sumcycle[j][k] < Lini+Ldev){
   RD[j][k][ii] = MinRD + (MaxRD - MinRD)*(Kcb[ii] - Kcb_ini)/(Kcb_mid - Kcb_ini);
if(RD[j][k][ii] > MaxRD) RD[j][k][ii] = MaxRD;
}
else{
 RD[j][k][ii] = MaxRD;
}
}
else if(smc_rz[j][k][ii-1] < WP_RZ[j][k][ii-1]) {
if(sumcycle[j][k] < Lini) Kcb[ii] = Ks[j][k][ii]*Kcb_ini;
if(sumcycle[j][k] >= Lini && sumcycle[j][k] < Lini+Ldev ){
Kcb[ii] = Ks[j][k][ii]*Kcb_ini + ((sumcycle[j][k] - Lini)/Lini)*(Ks[j][k][ii]*Kcb_mid –
```

```
        Ks[j][k][ii]*Kcb_ini);
}
if(sumcycle[j][k] >= Lini+Ldev && sumcycle[j][k] < Lini+Ldev+Lmid) Kcb[ii] =
        Ks[j][k][ii]*Kcb_mid;
        if(sumcycle[j][k] >= Lini+Ldev+Lmid && sumcycle[j][k] < Calendar){
                Kcb[ii] = Ks[j][k][ii]*Kcb_mid + (( sumcycle[j][k] -
                Lini+Ldev+Lmid))/Llate)*(Ks[j][k][ii]*Kcb_end - Ks[j][k][ii]*Kcb_mid);
        }
        if(sumcycle[j][k] < Lini+Ldev){
                 RD[j][k][ii] = MinRD + (MaxRD - MinRD)*(Kcb[ii] -
        Ks[j][k][ii]*Kcb_ini)/(Ks[j][k][ii]*Kcb_mid - Ks[j][k][ii]*Kcb_ini);
                 if(RD[j][k][ii] > MaxRD) RD[j][k][ii] = MaxRD;
        }
        else{
                RD[j][k][ii] = MaxRD;
        }
}//end of Kcb and RD adjust
//for fix loss
                irrig[j][k][ii] = sup_wrbd[j][k] * IFLAG[j][k][ii];
                if(ii == t_lag) {
                        if(IFLAG[j][k][ii] == 1) {
                                irrig[j][k][ii] = (float)(sup_wrbd[j][k] - fixloss) * IFLAG[j][k][ii];
                        }
                }
                if(ii > t_lag) {
                        if(IFLAG[j][k][ii-1] == 0 && IFLAG[j][k][ii] == 1) {
                                irrig[j][k][ii] = (float)(sup_wrbd[j][k] - fixloss) * IFLAG[j][k][ii];
                        }
                }
//new for sum up of fixloss
                if(ii > t_lag && ii < Calendar+t_lag){  //new!
                        if(IFLAG[j][k][ii-1] == 1 && IFLAG[j][k][ii] == 0) {
                                countn = 1;
                                Nfix[j][k][ii] = (float)(fixloss * countn);
                        }
                        else Nfix[j][k][ii] = 0.0f;
                }
                else if(ii == Calendar+t_lag-1){ //new!
                        if(IFLAG[j][k][ii-1] == 0 && IFLAG[j][k][ii] == 1){
                                countn = 1;
                                Nfix[j][k][ii] = (float)(fixloss * countn);
                        }
                        else Nfix[j][k][ii] = 0.0f;
                }
}// end of ii loop preparation of all input data
 // Now start of Main loop in every cycle
          for(ii = 0; ii < cycle[i][j][k]; ii++) {// Main LOOP for ii
        // Start to strip soil layer in calculation here //new
                nlr_RZ_MAX              = (int)(MaxRD/layerD); //no. of layers at RD=Max
                NLR_RZ[j][k][ii]= (int)(RD[j][k][ii]/layerD); // no. of layers at a time period
                RD[j][k][ii]        = (float)(NLR_RZ[j][k][ii]*layerD); //adjust RD by layer
                FC_RZ[j][k][ii]   = FC[j][k]*RD[j][k][ii]*1000;
                PWP_RZ[j][k][ii]= PWP[j][k]*RD[j][k][ii]*1000;
                RAM_RZ[j][k][ii]= (FC[j][k] - PWP[j][k])*(RD[j][k][ii]*1000)*P[ii];
                WP_RZ[j][k][ii]  = FC_RZ[j][k][ii] - RAM_RZ[j][k][ii]; //new for varying p
                if(ii > t_lag ){
                        if(NLR_RZ[j][k][ii] > NLR_RZ[j][k][ii-1]){
                                tsum[j][k][ii] = 0.0f;
```

```
                                for(nsl = NLR_RZ[j][k][ii-1]; nsl < NLR_RZ[j][k][ii]; nsl++){
                                        tsum[j][k][ii] = tsum[j][k][ii] + smc_layer[nsl][j][k][ii-1];
                                }
                                smc_rz[j][k][ii] = smc_rz[j][k][ii-1] + tsum[j][k][ii];
                        }
                        else smc_rz[j][k][ii] = smc_rz[j][k][ii-1];
                }
                // This is to sum up smc for global check
        if(ii == t_lag) sum_up_smc[j][k][ii] = WP_RZ[j][k][ii];
        else sum_up_smc[j][k][ii] = smc_rz[j][k][ii];
//now calculate supply in each t-period
        supply[j][k][ii] = irrig[j][k][ii];
        sumsupply[i] = sumsupply[i] + supply[j][k][ii] * subLat_area[j][k] / TotalAREA;  // new
//Soil Water Balance Main Equation
                if(ii < t_lag) {
                        Ea[j][k][ii] = 0.0f;
                        SMCline[j][k][ii] = 0.0f;
                }
                else if(ii==t_lag) {
                        Ea[j][k][ii] = ETc[j][k][ii]; //because initial smc = WP_RZ
                smc_rz[j][k][ii]  = (float)(WP_RZ[j][k][ii] + supply[j][k][ii] - Ea[j][k][ii] + ER[ii]);
                if(smc_rz[j][k][ii] < WP_RZ[j][k][ii]){
                        Ea[j][k][ii] = (smc_rz[j][k][ii] – PWP_RZ[j][k][ii]) * (ETc[j][k][ii]) /
                        (WP_RZ[j][k][ii]-PWP_RZ[j][k][ii]);

                smc_rz[j][k][ii] = (float)(WP_RZ[j][k][ii] + supply[j][k][ii] - Ea[j][k][ii] + ER[ii]);
                }
                }
                else if(ii > t_lag){
                if(smc_rz[j][k][ii] >= WP_RZ[j][k][ii]) {
                        Ea[j][k][ii] = ETc[j][k][ii];
                smc_rz[j][k][ii] = (float)(smc_rz[j][k][ii] + supply[j][k][ii] - Ea[j][k][ii] + ER[ii]);
                }
                else if(smc_rz[j][k][ii] < WP_RZ[j][k][ii]){
                Ea[j][k][ii] = (smc_rz[j][k][ii] -
                PWP_RZ[j][k][ii])*(ETc[j][k][ii])/(WP_RZ[j][k][ii]-PWP_RZ[j][k][ii]);
                smc_rz[j][k][ii] = (float)(smc_rz[j][k][ii] + supply[j][k][ii] - Ea[j][k][ii] + ER[ii]);
                }
                } // end Soil Moisture Balance Main Equation and its adjustment.
                SMCline[j][k][ii] = smc_rz[j][k][ii]; //for ploting
// this is when SMC > FC
                infilt = 0.0f;
                if(smc_rz[j][k][ii] > FC_RZ[j][k][ii]){
                        SMCline[j][k][ii] = FC_RZ[j][k][ii];
                        infilt = smc_rz[j][k][ii] - FC_RZ[j][k][ii];
                        infiltopup[j][k][ii] =  infilt;
                        smc_rz[j][k][ii] =  FC_RZ[j][k][ii];
                        for(nsl = NLR_RZ[j][k][ii]; nsl < nlr_RZ_MAX; nsl++) {
                                smc_layer[nsl][j][k][ii] = smc_layer[nsl][j][k][ii-1] + infilt;
                                if(smc_layer[nsl][j][k][ii] > FC_layer){
                                        infilt = smc_layer[nsl][j][k][ii] - FC_layer;
                                        smc_layer[nsl][j][k][ii] = FC_layer;
                                }
                        else {
                        infilt = 0.0;
                        break;
                                }
                        }
```

```
                    Ovr_supl[j][k][ii] = infilt;
                }
                else{
                    for(nsl = 0; nsl < nlr_RZ_MAX; nsl++) {
                        smc_layer[nsl][j][k][ii] = smc_layer[nsl][j][k][ii-1];
                    }
                    infiltopup[j][k][ii] = 0.0f;
                    Ovr_supl[j][k][ii] = 0.0f;
                }
                sumOversupl[i] = sumOversupl[i] + Ovr_supl[j][k][ii];
// This is for plotting smc line if oversupply
        if(Ovr_supl[j][k][ii] > 0.0f) SMCline[j][k][ii]  = FC_RZ[j][k][ii] + Ovr_supl[j][k][ii]; //new
// this is for calculation of Ks when smc_rz < WP_RZ
        if(smc_rz[j][k][ii] < WP_RZ[j][k][ii]) {
         Ks[j][k][ii] = (FC_RZ[j][k][ii] - smc_rz[j][k][ii])/(FC_RZ[j][k][ii]-WP_RZ[j][k][ii]);
        }
        else Ks[j][k][ii] = 1;
// this is a penalty function
        if(smc_rz[j][k][ii] < WP_RZ[j][k][ii]) {
        LWP[j][k][ii] = (float)(WP_RZ[j][k][ii] - smc_rz[j][k][ii]);
        Equity[j][k][ii] = (float)pow((R1*LWP[j][k][ii]),2);
//this is for plotting
        SMCline[j][k][ii] = WP_RZ[j][k][ii] - LWP[j][k][ii];
}
else {
        LWP[j][k][ii] = 0.0f;
        Equity[j][k][ii] = 0.0f;
}
        sumLWP[i] = sumLWP[i] + LWP[j][k][ii];
        sumEquity[i] = sumEquity[i] + Equity[j][k][ii];
// This is to compute Global Balance Check
        sumsmc_rz[i][j][k]          = sumsmc_rz[i][j][k] + sum_up_smc[j][k][ii];
        sumX[i][j][k]               = sumX[i][j][k]     + supply[j][k][ii];
        sumETc[i][j][k]      = sumETc[i][j][k]    + ETc[j][k][ii];
        sumEa[i][j][k]       = sumEa[i][j][k]     + Ea[j][k][ii];
        sumSMCend[i][j][k]   = sumSMCend[i][j][k] + smc_rz[j][k][ii];
        sumO[i][j][k]        = sumO[i][j][k]      + Ovr_supl[j][k][ii];
        cum_stress[i][j][k]  = cum_stress[i][j][k]+ LWP[j][k][ii];
        sumNfix[i][j][k]     = sumNfix[i][j][k]   + Nfix[j][k][ii];
        suminfil[i][j][k]    = suminfil[i][j][k]  + infiltopup[j][k][ii];
        GlobalSMC[i][j][k]   = sumsmc_rz[i][j][k] + sumX[i][j][k] - sumEa[i][j][k] - suminfil[i][j][k]
- sumSMCend[i][j][k];
        sumIrr_day[i][j][k]  = sumIrr_day[i][j][k] + IFLAG[j][k][ii];
}// end of ii Main loop
}// end of  k
}// end of j main loop

//This is for capacity constraint in Branch and Main canal
        for(j = 0; j <NSCHEME; j++){//Bcapa constraint is computed here.
        for(ii=0; ii<CYCLE; ii++){  //new
        sumTertiaryFlow[j][ii] = 0.0f;
          for(k = 0; k <num_sublat[j]; k++){
                S_latGO[ii][k] = Scapa[j][k]*IFLAG[j][k][ii];
                    sumTertiaryFlow[j][ii]  = sumTertiaryFlow[j][ii] + S_latGO[ii][k];
                }
        if(sumTertiaryFlow[j][ii] > Bcapa[j]) {
                capaConstr[j][ii] = (float)R2*(sumTertiaryFlow[j][ii] - Bcapa[j]);
        }
```

```
        else capaConstr[j][ii] = 0.0f;
        sumBcapaCon[i] =  sumBcapaCon[i] + (float)pow(capaConstr[j][ii],2);  //CONSTRAINTS
}       // end of cycle ii
} //end j
//new transfer IFLAG to IFLAG_B
        for(ii=0; ii<CYCLE; ii++) {
                for(j = 0; j <NSCHEME; j++) {  //Bcapa constraint is computed here
                        sumIFLAG[j][ii] = 0;
                        for(k = 0; k <num_sublat[j]; k++){
                                sumIFLAG[j][ii] = sumIFLAG[j][ii] + IFLAG[j][k][ii];
                        }
                                if(sumIFLAG[j][ii] > 0) IFLAG_B[j][ii] = 1;
                                else IFLAG_B[j][ii] = 0;
                }
        }
//Main capa. constraint starts here.
        for(ii=0; ii<CYCLE; ii++){ //new              //added  new
                sumSecondaryFlow[ii] = 0.0f;
        for(j = 0; j <NSCHEME; j++){
                 B_GO[ii][j] = Bcapa[j]*IFLAG_B[j][ii];
                 sumSecondaryFlow[ii] = sumSecondaryFlow[ii] + B_GO[ii][j];
        }
        }
        for(ii=0; ii<CYCLE; ii++){ //new
        if(sumSecondaryFlow[ii] > (main_capa*PercentFlow/100)){
                MnCapaCon[ii] = R3*(sumSecondaryFlow[ii] - main_capa*PercentFlow/100);
        }
                else MnCapaCon[ii] = 0.0f;
                        sumMCapaCon[i] = sumMCapaCon[i] + (float)pow(MnCapaCon[ii],2);
                }
ftnss[i] = (float)(sumsupply[i] + sumEquity[i] + sumBcapaCon[i] + sumMCapaCon[i]);
sumftnss = sumftnss + ftnss[i];
if(bestfit  > ftnss[i]){
        bestfit         = ftnss[i];
        psumLWP         = sumLWP[i];
        psumOVR         = sumOversupl[i];
        psumBcapa       = sumBcapaCon[i];
        psumMCapaCon    = sumMCapaCon[i];
        psumsupply      = sumsupply[i];
        psumEquity      = sumEquity[i];
                for(j = 0; j<NSCHEME; j++) {
                        for(k = 0; k < num_sublat[j]; k++) {
                                pwara[j][k]     = wara[j][k];
                                pshut[j][k] = shut[j][k];
                                pstrt[j][k] = strt[j][k];
                                psumX[j][k] = sumX[i][j][k];
                                psumD[j][k] = sumD[i][j][k];
                                psumsmc_rz[j][k] = sumsmc_rz[i][j][k];
                                psuminfil[j][k] = suminfil[i][j][k];
                                psumEa[j][k] = sumEa[i][j][k];
                                psumETc[j][k] = sumETc[i][j][k];
                                psumSMCend[j][k] = sumSMCend[i][j][k];
                                psumNfix[j][k] = sumNfix[i][j][k];
                                psumO[j][k] = sumO[i][j][k];
                                pstress[j][k] = cum_stress[i][j][k];
                                pGlobalSMC[j][k] = GlobalSMC[i][j][k];
                                psumIrr_day[j][k] = sumIrr_day[i][j][k];
                                pcycle[j][k]    = cycle[i][j][k];
```

```
for(ii = 0; ii<CYCLE; ii++) {
        pFlag[j][k][ii] = IFLAG[j][k][ii];
        pETc[j][k][ii] = ETc[j][k][ii];
        pEa[j][k][ii] = Ea[j][k][ii];
        psupply[j][k][ii] = supply[j][k][ii];
        pdemand[j][k][ii] = demand[j][k][ii];
        psmc_rz[j][k][ii] = smc_rz[j][k][ii];
        pSMCline[j][k][ii] = SMCline[j][k][ii];
        pOvr_supl[j][k][ii] = Ovr_supl[j][k][ii];
        pLWP[j][k][ii] = LWP[j][k][ii];
        ptsum[j][k][ii] = tsum[j][k][ii];
        prnWP_RZ[j][k][ii] = WP_RZ[j][k][ii];
        prnPWP_RZ[j][k][ii] = PWP_RZ[j][k][ii];
        pFC_RZ[j][k][ii] = FC_RZ[j][k][ii];
}//end ii
}//end of k sublat
        for(ii=0; ii<CYCLE; ii++ ){
         pcapaConstr[j][ii] = capaConstr[j][ii];
         pFlag_B[j][ii] = IFLAG_B[j][ii];
        }
}//end of j
}// end of if all constraints are satisfied
}//end of i

if (gen == MaxGEN-1 || DELTA_ftnss <= DELTA ) {
index = 999;
for(j = 0; j<NSCHEME; j++) {
fprintf(f1,"IFLAG_S  ");
for(ii=0; ii<CYCLE; ii++)  fprintf(f1,"%9d",pFlag_B[j][ii]); fprintf(f1,"\n");
 for(k = 0; k < num_sublat[j]; k++) {
 N_fixloss = (float)(psumNfix[j][k]/fixloss);
 fprintf(f1,"Irrifrq%3.0f\n", N_fixloss);
 fprintf(f1,"Secondary%2d Tertiary%2d Qflow result = %5.3f m3/s \n", j+1, k+1, Scapa[j][k]);
 fprintf(f1,"IFLAG    ");
for(ii=0; ii<pcycle[j][k]; ii++)  fprintf(f1,"%9d",pFlag[j][k][ii]);
fprintf(f1, " Irri_day   %9.3f sumFixloss   %9.3f\n", psumIrr_day[j][k], psumNfix[j][k]);
fprintf(f1,"FC       ");
        for(ii=0; ii<pcycle[j][k]; ii++)  fprintf(f1,"%9.3f",pFC_RZ[j][k][ii]); fprintf(f1,"\n");
        fprintf(f1,"tsum     ");
        for(ii=0; ii<pcycle[j][k]; ii++)  fprintf(f1,"%9.3f",ptsum[j][k][ii]); fprintf(f1,"\n");
        fprintf(f1,"Supply   ");
        for(ii=0; ii<pcycle[j][k]; ii++)  fprintf(f1,"%9.3f",psupply[j][k][ii]); fprintf(f1,"\n");
        fprintf(f1,"ETc      ");
        for(ii=0; ii<pcycle[j][k]; ii++)  fprintf(f1,"%9.3f",pETc[j][k][ii]); fprintf(f1,"\n");
        fprintf(f1,"Ea       ");
        for(ii=0; ii<pcycle[j][k]; ii++)  fprintf(f1,"%9.3f",pEa[j][k][ii]); fprintf(f1,"\n");
        fprintf(f1,"smc_rz   ");
        for(ii=0; ii<pcycle[j][k]; ii++)  fprintf(f1,"%9.3f",psmc_rz[j][k][ii]);fprintf(f1,"\n");
        fprintf(f1,"SMC      ");
        for(ii=0; ii<pcycle[j][k]; ii++)  fprintf(f1,"%9.3f",pSMCline[j][k][ii]); fprintf(f1,"\n");
        fprintf(f1,"WP       ");
        for(ii=0; ii<pcycle[j][k]; ii++)  fprintf(f1,"%9.3f",prnWP_RZ[j][k][ii]); fprintf(f1,"\n");
        fprintf(f1,"PWP      ");
        for(ii=0; ii<pcycle[j][k]; ii++)  fprintf(f1,"%9.3f",prnPWP_RZ[j][k][ii]); fprintf(f1,"\n");
        fprintf(f1,"BelowWP  ");
        for(ii=0; ii<pcycle[j][k]; ii++)  fprintf(f1,"%9.3f",pLWP[j][k][ii]);
        fprintf(f1, " cum_stress %9.3f\n", pstress[j][k]);
        fprintf(f1,"Global  %9.3f\n", pGlobalSMC[j][k]);
```

```
                    fprintf(f1,"SMCstr  %9.3f\n", psumsmc_rz[j][k]);
                    fprintf(f1,"SMCend  %9.3f\n", psumSMCend[j][k]);
                    fprintf(f1,"Irriga  %9.3f\n", psumX[j][k]);
                    fprintf(f1,"ETc     %9.3f\n", psumETc[j][k]);
                    fprintf(f1,"Ea_     %9.3f\n", psumEa[j][k]);
                    fprintf(f1,"infilt  %9.3f\n", psuminfil[j][k]);
                    fprintf(f1,"Drainage%9.3f\n", psumO[j][k]);
                    fprintf(f1,"stress  %9.3f\n", pstress[j][k]);
                    } // end of if k
} //end of if j
fprintf(f1," %4.2f  %4.3f", PXOVER, NofMu);
fprintf(f1," %4d %7.3f %7.3f %2d %7.3f %2d  R2P2 %7.3f R3 %2d R3P3 %7.3f\n",
        gen, ftnss[POPSIZE],  psumsupply, R1, psumEquity, R2, psumBcapa, R3, psumMCapaCon);
}// end of if loop
        avgftnss = sumftnss/POPSIZE;
        minftnss  = ftnss[0];
        for(i=0; i<POPSIZE; i++) {
                if(ftnss[i]  < minftnss)   minftnss  = ftnss[i];
                if(ftnss[i]  < ftnss[POPSIZE]) ftnss[POPSIZE] = ftnss[i];
        }
fprintf(f1,"%3d %9.2f\n", gen+1, ftnss[POPSIZE]);
}
/*********************************/
void Tournament() {
        int ii,jj,j,h,k,mem;
        for(mem=0; mem < POPSIZE; mem++){
                ii = RandInt(0,POPSIZE-1);
                jj = RandInt(0,POPSIZE-1);
                if(ftnss[ii] <= ftnss[jj]){
                for(j=0; j<NSCHEME; j++){
                  for(k = 0; k < num_sublat[j]; k++) {
                        for(h=0; h<LENGTH; h++){
                                mated_pop[mem][j][k][h] = initial_pop[ii][j][k][h];
                        }
                  }
                }
                }
     if(ftnss[ii] > ftnss[jj]){
                        for(j=0; j<NSCHEME; j++){
                          for(k = 0; k < num_sublat[j]; k++) {
                                for(h=0; h<LENGTH; h++){
                                        mated_pop[mem][j][k][h] = initial_pop[jj][j][k][h];
                                }
                          }
                        }
                }
        }
}// end of tournament
/*********************************/
// Routine to crossover population
void XOVER(int i1, int i2) {
        int j, h, k, kk;
        kk = RandInt(1,LENGTH-1);
  for(j = 0; j<NSCHEME; j++){
          for(k = 0; k < num_sublat[j]; k++) {
                for(h =kk; h<LENGTH; h++)  {
                  SwapRealValues(&mated_pop[i1][j][k][h],&mated_pop[i2][j][k][h]);
                }
```

```
            }
    }
}
/*********************************/
void UNIXOVER(int i1, int i2) {
int j, h, k, kk;
  for(j= 0; j<NSCHEME; j++){
   for(k = 0; k < num_sublat[j]; k++) {
            for(h = 0; h<LENGTH; h++)  {
                        kk= rand()&01;
                        if(kk==1) SwapRealValues(&mated_pop[i1][j][k][h],&mated_pop[i2][j][k][h]);
            }
    }
   }
}
/*************************************/
// Routine to crossover population, 2-site
void TPXOVER(int i1, int i2) {
            int j,h,k,kk1,kk2;
            kk1 = RandInt(1,LENGTH-1);
            kk2 = RandInt(1,LENGTH-1); // this is for 2-site XOVER
            if (kk1>kk2) SwapIntValues(&kk1,&kk2); // this is to swap 2-site if the later is less
  for(j= 0; j<NSCHEME; j++){
   for(k = 0; k < num_sublat[j]; k++) {
            for(h=kk1;h<kk2;h++)  {
                        SwapRealValues(&mated_pop[i1][j][k][h],&mated_pop[i2][j][k][h]);
    }
   }
  }
}
/*************************************/
void SelectMate() {
int  j, k, h, mem, num_select, one;
float Prob_X;
num_select = 0;
            for (mem = 0; mem< POPSIZE ; mem++) {
                        Prob_X = rand()%1000/1000.0f;
                        if (Prob_X < PXOVER){
                                    ++num_select;
                                    if(num_select%2 == 0) XOVER(one,mem);
                                    else one = mem;
                        }
            }
}
/*****************************/
void UniSelectMate() {
int   j, k, h, mem, num_select,one;
float Prob_X;
   num_select = 0;
            for (mem = 0; mem< POPSIZE ; mem++) {
                        Prob_X = rand()%1000/1000.0f;
                        if (Prob_X < PXOVER){
                                    ++num_select;
                                    if(num_select%2 == 0) UNIXOVER(one,mem);
                                    else one = mem;
                        }
            }
}
```

```
/*****************************************/
// this is to radom probability of mutation
void mutatepop() {
  int i,j,k,h,mutant;
  double r;
  for (i=0;i<POPSIZE;i++) {
          for(j=0; j<NSCHEME; j++){
           for(k = 0; k < num_sublat[j]; k++) {
                   r = rand()%1000/1000.0;
                   if(r<Pmutate){
                   mutant = rand()&01;
                   if(mutant == 1) mated_pop[i][j][k][0] = (float)(mated_pop[i][j][k][0] + inc);
                   else mated_pop[i][j][k][0] = (float)(mated_pop[i][j][k][0] - inc);
                   }
                   if(mated_pop[i][j][k][0] < MinLag) mated_pop[i][j][k][0] = (float)(MinLag);
                   else if(mated_pop[i][j][k][0] > MaxLag) mated_pop[i][j][k][0] = (float)(MaxLag);
          for(h= 1; h<LENGTH; h++){
                   r = rand()%1000/1000.0;
                   if(r<Pmutate){
                   mutant = rand()&01;
                   if(mutant == 1) mated_pop[i][j][k][h] = (float)(mated_pop[i][j][k][h] + inc);
                   else mated_pop[i][j][k][h] = (float)(mated_pop[i][j][k][h] - inc);
                   }
          if(mated_pop[i][j][k][h] < Min_wrbd) mated_pop[i][j][k][h] = (float)(Min_wrbd);
          else if(mated_pop[i][j][k][h] > Max_wrbd) mated_pop[i][j][k][h] = (float)(Max_wrbd);
          }
          }
          }
          for(j=0;j<NSCHEME;j++)  {
            for(k = 0; k < num_sublat[j]; k++) {
              for(h = 0; h<LENGTH; h++)  initial_pop[i][j][k][h] = mated_pop[i][j][k][h];
            }
          }
}//end of i POPSIZE
/*****************************************/
// Routine to print error message
void nrerror(char *messg) {
          puts(messg) ;
          exit(1) ;
}
/*****************************************/
// Routine to swop value
void SwapIntValues(int *x, int *y) {
          int temp;
          temp = *x;
          *x=*y;
          *y=temp;
}
/*****************************************/
// Routine to swop real value
void SwapRealValues(float *xx, float*yy) {
          float temp;
          temp = *xx;
          *xx = *yy;
          *yy = temp;
}
/*****************************************/
```

//GA47A.c Program to formulate GAs for water scheduling problem

```
//      Water Scheduling for Pugal system (zero-1 criteria)
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <process.h>
#include <time.h>

#define Tperiod         1.0     // input time period (0.5 or 1.0)
#define POPSIZE         100     // input population size
#define NSCHEME         15      //input number of scheme
#define Calendar        195     //input total growth period
#define MaxLag          15      //input maximum staggering day from the start
#define MinLag          0       //input minimum stagerring day from the start
#define STAGE           195     //input total stage day
#define CYCLE           210     //input total time step (CYCLE+MaxLag)
#define LENGTH          CYCLE+1 //195 + possible lag days (15) + 1 gene for starting Tperiod
#define inc             Tperiod //input increment of mutation
#define INTERVAL        300     // input interval of generation to recheck of improvement fitness
#define DELTA           0.0001  // input minimum improvement of fitness after INTERVAL
                        generations
#define IrriEff         0.9     // input irrigation efficiency
#define  layerD         0.01    // input a root zone soil layer for water balance in meter
#define  tDepth         1.5     // input total depth of soil layer in meter
#define MaxLyr          150     // input maximum soil layer in consider

int     RandInt (int, int);
float   RandReal (float,float);
void    Read_inputfiles();
void    SwapIntValues(int *, int *);
void    SwapRealValues(float *, float *);
void    nrerror(char *messg) ;
void    initial();
void    objective(int);
void    SelectMate();
void    UniSelectMate();
void    mutatepop();
void    Tournament();

int     i, tt, index, mated[POPSIZE], t_lag[NSCHEME];
float   mated_pop[POPSIZE][NSCHEME][LENGTH],
        initial_pop[POPSIZE][NSCHEME][LENGTH],Pmutate;
float   ftnss[POPSIZE+1], DELTA_ftnss, OLD_minftnss, bestfit;
int     IFLAG[NSCHEME][CYCLE],  pFlag[NSCHEME][CYCLE];
int     MaxGEN, R1, R2, PercentFlow;
float   PXOVER, NofMu;
float   countday;
int     root_period, stage_day[STAGE], stagenum[STAGE];
float   D_factr;
float   MinRD, MaxRD, Lini, Ldev, Lmid, Llate, Kcb_ini, Kcb_mid, Kcb_end, Kcb[CYCLE];
float   FC[NSCHEME], PWP[NSCHEME];
float   Bflowrate[NSCHEME], input_ETc[MaxLag][STAGE], main_capa;
float   scheme_area[NSCHEME], tot_area;
float   psumOVR, psumLWP, psumCapa;
float   sumOversupl[POPSIZE], sumCapaConstr[POPSIZE], sumLWP[POPSIZE];
```

```
float  LWP[NSCHEME][CYCLE], pLWP[NSCHEME][CYCLE];
float  Ovr_supl[NSCHEME][CYCLE], pOvr_supl[NSCHEME][CYCLE];
float  ETc[NSCHEME][CYCLE], pETc[NSCHEME][CYCLE];
float  Eff_rain[MaxLag][STAGE], fixloss[NSCHEME], fxloss[NSCHEME][CYCLE];
float  sup_wrbd[NSCHEME],      psup_wrbd[NSCHEME][CYCLE];
float  supply[NSCHEME][CYCLE],                 psupply[NSCHEME][CYCLE];
float  demand[NSCHEME][CYCLE],                 pdemand[NSCHEME][CYCLE];
float  capaConstr[POPSIZE][CYCLE],                     pcapaConstr[CYCLE];
int    pstrt[NSCHEME], strt[NSCHEME];

FILE *f1,*f2,*f3 ;

//MAIN PROGRAM
void main (void) {
   int gen;
   clock_t start, finish;
   float  duration, TotalTime ;
   TotalTime = 0.0f;
   if(!(f1 = fopen("ga47Aper.out","w"))) {
                printf("can't open output file\n");
                exit(1);
        }
   Read_inputfiles();
        srand(2);
        start = clock();
   MaxGEN      = 2500;
        R1              = 1;
        R2              = 10000;
        PXOVER          = 0.9f;
        NofMu  = 0.09f;
// R1   = 1;
//do{
// R2   = 10000;
//do{
//      NofMu  = 0.12f;
//do{
// PXOVER      = 0.5f;
//do{
//PercentFlow = 85;
//do{
   initial();
   index = 0;
   OLD_minftnss = 1000.0f;
        DELTA_ftnss             = 1000.0f;
        bestfit                 = 10000000000.0f;
        ftnss[POPSIZE]  = 10000000000000.0f;
 for(gen = 0; gen < MaxGEN; gen++) {
        objective(gen); if(index == 999) break;
        Tournament();
//      SelectMate();
        UniSelectMate();
        mutatepop();
   if (!((gen+1)%INTERVAL)){
     DELTA_ftnss = (float)fabs((double)((OLD_minftnss - ftnss[POPSIZE])/ftnss[POPSIZE]));
     OLD_minftnss = ftnss[POPSIZE];
}
 } //End of Generation Loop
```

```
//  PercentFlow = PercentFlow + 5;
//}while (PercentFlow <= 91);
//  PXOVER = (float)(PXOVER + 0.05f);
//}while (PXOVER <= 0.85f);
// NofMu = (float)(NofMu + 0.03);
//}while (NofMu  <= 0.16);
//R2 = R2 + 2500;
//}while (R2 <= 17501);
// R1 = R1 + 2;
//}while (R1 <= 10);
  finish = clock();
  duration = (float)(finish - start) / CLOCKS_PER_SEC;
  TotalTime = (float)(TotalTime + duration);
  (void) fclose (f1);
} //MAIN PROGRAM END
/*********************/
void Read_inputfiles() {
   int   ii, jj, kk, num_schemes, scheme_node[NSCHEME];
         char  title[80],title2[80],title3[80],title4[80],title5[80],title6[80],title7[80],title8[80];
         char  title9[80], title10[80], title11[80], title12[80], schedulefile[20];
         if(!(f2 = fopen("Proj47A.lst","r"))) {   //for ga47A.dat
    nrerror(" Can not open project file");
   }
   // now read the data file names
         fgets(schedulefile, 20, f2) ;
         for(ii = 0; ii < 20; ii++) {
                  if(schedulefile[ii] == '\n') schedulefile[ii] = '\0' ;
         }
         (void) fclose(f2) ;
         if(!(f2 = fopen(schedulefile,"r"))) {
    nrerror(" Unable to open schedulefile") ;
   }
   fgets(title, 80, f2);            // Water scheduling title
   fscanf(f2,"%d",&num_schemes);  // scan number of schemes here
   for(jj=0;jj<num_schemes;jj++) fscanf(f2,"%d ", &scheme_node[jj]);
   fgets(title2, 80, f2);        // scheme area (ha) title
   for(jj=0;jj<num_schemes;jj++) fscanf(f2,"%f ", &scheme_area[jj]);
         tot_area = 0.0f;
         for(jj=0;jj<num_schemes;jj++){
                  tot_area = tot_area + scheme_area[jj];  // new
         }
   fgets(title3, 80, f2);        // main canal supply title
         fscanf(f2,"%f ", &main_capa) ;  // scan main canal supply
   fgets(title4, 80, f2);        // branch canal title
         for(jj = 0; jj < num_schemes; jj++) fscanf(f2,"%f ", &Bflowrate[jj]) ;
         fscanf(f2,"\n");
   fgets(title5, 80, f2);        // CWR title
   for(tt=0; tt<STAGE; tt++) fscanf(f2,"%d ", &stagenum[tt]);
   for(tt=0; tt<STAGE; tt++) fscanf(f2,"%d ", &stage_day[tt]);
   for(kk = 0; kk < MaxLag; kk++) {
                  for(tt = 0; tt <STAGE; tt++){
                    fscanf(f2,"%f", &input_ETc[kk][tt]) ;
                  }
         }
   fscanf(f2,"\n");
   fgets(title6, 80, f2);        // ER title
         for(kk = 0; kk< MaxLag; kk++){
                  for(tt = 0; tt <STAGE; tt++){
```

```
                        fscanf(f2,"%f", &Eff_rain[kk][tt]) ;
                    }
            }
    fscanf(f2,"\n");
            fgets(title7, 80, f2);          // loss factor title
            for(jj = 0; jj < num_schemes; jj++) fscanf(f2,"%f ", &fixloss[jj]) ;
            fscanf(f2,"\n");
            for(ii=0; ii<CYCLE; ii++){
             for(jj = 0; jj < num_schemes; jj++) fxloss[jj][ii] = fixloss[jj];
            }
    fgets(title8, 80, f2);          // field capacity title
            for(jj = 0; jj < num_schemes; jj++) {
                    fscanf(f2,"%f ", &FC[jj]) ;
            }
    fgets(title9, 80, f2);          // permanent wilting point title
            for(jj = 0; jj < num_schemes; jj++) {
                    fscanf(f2,"%f ", &PWP[jj]) ;
            }
    fscanf(f2,"\n");
            fgets(title10,80,f2);    // Length stage title
            fscanf(f2,"%f %f %f %f",&Lini, &Ldev, &Lmid, &Llate);
    fscanf(f2,"\n");
            fgets(title11,80,f2);    // Kcb title
            fscanf(f2,"%f %f %f",&Kcb_ini, &Kcb_mid, &Kcb_end);
    fscanf(f2,"\n");
    fgets(title12, 80, f2);  // rooting depth title
            fscanf(f2,"%f %f", &MinRD, &MaxRD);
    fscanf(f2,"\n");
            fscanf(f2,"%f",&D_factr);
    (void)fclose(f2);
}
/*********/
float RandReal (float lower,float upper){
            float val;
            float nb = ( upper - lower );
            val =  (float)(rand()%1000/1000.0*nb + lower);
            return ((float)val);
}
/*********/
int RandInt (int lower,int upper){
            int val;
            int nb = (upper - lower );
            val =  (rand()%nb + lower);
            return val;
}
/*********/
void initial(){
  int       i,j,jj;
  float nd = RAND_MAX;
  for(i=0;i<POPSIZE;i++) {
              for(jj = 0; jj < NSCHEME; jj++){
                      initial_pop[i][jj][0] = (float)(MinLag + inc *(float)floor((1+((MaxLag-1) -
MinLag)/inc)*(float)rand()/(nd+1.00)));
                      for(j=1;j<LENGTH;j++)  initial_pop[i][jj][j] = (float)(rand()&01);
              }
  }
/**********/
```

```
void objective(int gen){
        int             i, k, jj, ii, cycle[POPSIZE][NSCHEME], cnvrs, pcycle[NSCHEME];
        float   sumftnss, minftnss, avgftnss;
        float   sumcycle, odd2[POPSIZE][STAGE];
        float   branchGO[CYCLE][NSCHEME], sumBflow[CYCLE], odds[POPSIZE];
        float   GlobalSMC[POPSIZE][NSCHEME],pGlobalSMC[NSCHEME];
        float   sumX[POPSIZE][NSCHEME], psumX[NSCHEME];
        float   sumsmc_rz[POPSIZE][NSCHEME], psumsmc_rz[NSCHEME];
        float   smc_rz[NSCHEME][CYCLE], psmc_rz[NSCHEME][CYCLE];
        float   sumsupply[POPSIZE], sumGlobal[POPSIZE], sumO[POPSIZE][NSCHEME];
        float   psumsupply, psumO[NSCHEME];
        float   psumEquity, sumEquity[POPSIZE], Equity[NSCHEME][CYCLE];
        float   RD[NSCHEME][CYCLE];
        float   pSMCline[NSCHEME][CYCLE], SMCline[NSCHEME][CYCLE];
        float   P[CYCLE], Ks[NSCHEME][CYCLE];
        int             NLR_RZ[NSCHEME][CYCLE], nlr_RZ_MAX, nsl;
        float   FC_layer, RAM_layer, WP_layer, RAM_RZ[NSCHEME][CYCLE];
        float   FC_RZ[NSCHEME][CYCLE], PWP_RZ[NSCHEME][CYCLE],
WP_RZ[NSCHEME][CYCLE];
        float   pFC_RZ[NSCHEME][CYCLE], prnPWP_RZ[NSCHEME][CYCLE],
prnWP_RZ[NSCHEME][CYCLE];
        float   smc_layer[200][NSCHEME][CYCLE], tsum[NSCHEME][CYCLE],
ptsum[NSCHEME][CYCLE], infilt ; //new
        float   irrig[NSCHEME][CYCLE], pstress[NSCHEME],
cum_stress[POPSIZE][NSCHEME];
        float   pRD[NSCHEME][CYCLE];
        float   psuminfil[NSCHEME], suminfil[POPSIZE][NSCHEME];
        float   pinfiltopup[NSCHEME][CYCLE], infiltopup[NSCHEME][CYCLE];
        float   ER[NSCHEME][CYCLE];
        float   Ea[NSCHEME][CYCLE], pEa[NSCHEME][CYCLE],
sumEa[POPSIZE][NSCHEME],psumEa[NSCHEME];
        float   sumIrr_day[POPSIZE][NSCHEME], psumIrr_day[NSCHEME],
sum_up_smc[NSCHEME][CYCLE];
        float   sumSMCend[POPSIZE][NSCHEME], psumSMCend[NSCHEME];
        float   N_fixloss, netinfil[NSCHEME][CYCLE], sumnetinfil[POPSIZE][NSCHEME],
psumnetinfil[NSCHEME];
        float   sumETc[POPSIZE][NSCHEME], psumETc[NSCHEME];
        float   cnvrs_ETc[NSCHEME][STAGE], cnvrs_rain[NSCHEME][STAGE],
pEaETcRatio[NSCHEME];
        float   sumER[POPSIZE][NSCHEME], psumER[NSCHEME],psumLwirr[NSCHEME];
        float   acumloss[CYCLE], xx, sumLwirr[POPSIZE][NSCHEME],
lwirri[NSCHEME][CYCLE];

 sumftnss = 0.0f;
for(i=0;i<POPSIZE;i++) {
 ftnss[i] = 0.0f;
 sumLWP[i] = 0.0f;
 sumOversupl[i] = 0.0f;
 sumCapaConstr[i] = 0.0f;
 sumGlobal[i] = 0.0f;
 sumsupply[i] = 0.0f;
 sumEquity[i] = 0.0f;
  for(jj = 0; jj < NSCHEME; jj++) {
        strt[jj] = (int)(initial_pop[i][jj][0]);
  t_lag[jj] = strt[jj];
        cycle[i][jj] = (int)(Calendar+ t_lag[jj]);
  sup_wrbd[jj] = (float)(IrriEff*Bflowrate[jj]*Tperiod*3600*24*1000)/(scheme_area[jj]*10000);
 }
```

```
// new transfer cnvrs_ETc and cnvrs_rain by t_lag[jj]
  for(jj = 0; jj < NSCHEME; jj++) { //new
          cnvrs = (int)(t_lag[jj]);
          for(tt = 0; tt< STAGE; tt++){
                  cnvrs_ETc[jj][tt] = input_ETc[cnvrs][tt];
                  cnvrs_rain[jj][tt] = Eff_rain[cnvrs][tt];
          }
  } // end new transfer array
  odds[i]  = (float)(Calendar - (Tperiod*(int)floor(Calendar/Tperiod) ) );
  Pmutate  = (float)(NofMu/(LENGTH*NSCHEME));          //new
  for(tt = 0; tt < STAGE; tt++){
           if(tt == 0) {
                  odd2[i][tt] = (float)( (stage_day[tt] - ((int)floor(stage_day[tt]/Tperiod))*Tperiod) );

           }
           else{
                  odd2[i][tt] = (float)( (odd2[i][tt-1] + stage_day[tt] - ((int)floor((odd2[i][tt-
1]+stage_day[tt])/Tperiod))*Tperiod) );
           }
  }
// Transfer decision variables here!
          for(jj = 0; jj < NSCHEME; jj++){
                  k = 1;
                  for(ii = 0; ii<cycle[i][jj]; ii++){
                          if(ii < t_lag[jj]) IFLAG[jj][ii] = 0;
                          if(ii >=  t_lag[jj]) {
                                  IFLAG[jj][ii] = (int)(initial_pop[i][jj][k]);
                                  k++;
                          }
                  }
          }
   for(jj = 0; jj <NSCHEME; jj++) { //Main jj loop
           sumX[i][jj] = 0.0f;
           sumEa[i][jj] = 0.0f;
           sumETc[i][jj] = 0.0f;
           suminfil[i][jj] = 0.0f;
           sumO[i][jj] = 0.0f;
           sumsmc_rz[i][jj] = 0.0f;
     cum_stress[i][jj] = 0.0f;
           sumIrr_day[i][jj] = 0.0f;
           sumSMCend[i][jj] = 0.0f;
           sumnetinfil[i][jj] = 0.0f;
           sumER[i][jj] = 0.0f;
           sumLwirr[i][jj] = 0.0f;
           tt = 0;
           sumcycle = 0.0f;
           countday = (float)(stage_day[0]);
     for(ii = 0; ii<cycle[i][jj]; ii++){// Sub LOOP for ii
//new transfer ETc from(mm/period)
                  if(ii < t_lag[jj]) {
                          ETc[jj][ii] = 0.0f;
                          ER[jj][ii] = 0.0f;  //new
                  }
                  else if(ii >= t_lag[jj]) {
                    sumcycle = (float)(sumcycle + Tperiod);
                    if(sumcycle <= countday){ //new adjust
         ETc[jj][ii] = (float)( ( cnvrs_ETc[jj][tt]/stage_day[tt]) *Tperiod);
         ER[jj][ii] = (float)( (cnvrs_rain[jj][tt]/stage_day[tt]) *Tperiod);
```

```
                    }
                    else{
                            if(stagenum[tt] != STAGE){
                                    ETc[jj][ii] = (float)( ( (cnvrs_ETc[jj][tt])/stage_day[tt])
*odd2[i][tt] + (Tperiod - odd2[i][tt]) * (cnvrs_ETc[jj][tt+1])/stage_day[tt+1] ) ; //*odds days
                                    ER[jj][ii] = (float)( ( (cnvrs_rain[jj][tt])/stage_day[tt]) *odd2[i][tt]
+ (Tperiod - odd2[i][tt]) * (cnvrs_rain[jj][tt+1])/stage_day[tt+1] ) ; //*odds days
                            }
                            else {
                                    ETc[jj][ii] = (float)( ( (cnvrs_ETc[jj][tt])/stage_day[tt]) *
odd2[i][tt]);
                                    ER[jj][ii] = (float)( ( (cnvrs_rain[jj][tt])/stage_day[tt]) *
odd2[i][tt]);
                            }
                       tt++;
                       countday = countday + stage_day[tt];
                    }
                   }
                   // This is to adjust D_factor  (Allen et al. 1998)
                   if(ETc[jj][ii] > 5.0f){        //new move
                            P[ii] = (float)(D_factr + 0.04*(5 - ETc[jj][ii]));
                            if(P[ii] < 0.1) P[ii] = 0.1f;
                            if(P[ii] > 0.8) P[ii] = 0.8f;
                   }
                   else P[ii] = D_factr;
//Set up smc in every soil strip layer as wilting point of every soil layer
                   FC_layer        = (float)(FC[jj]*layerD*1000);      //new all loop
                   RAM_layer       = (float)((FC[jj] - PWP[jj])*layerD*1000*P[ii]);
                   WP_layer        = FC_layer - RAM_layer;

                   for(nsl = 0; nsl < MaxLyr; nsl++) {
                            smc_layer[nsl][jj][ii-1] = WP_layer;
                   }
// For Kcb adjust and then cal of RD adjust
                   if(ii < t_lag[jj]) RD[jj][ii] = 0.0f;
                   else if(ii >= t_lag[jj] || smc_rz[jj][ii-1] >= WP_RZ[jj][ii-1]) { //new
                     if(sumcycle < Lini) Kcb[ii] = Kcb_ini; // new (0-24)
                     if(sumcycle >= Lini && sumcycle < Lini+Ldev ){ //(25-49)
                              Kcb[ii] = Kcb_ini + ((sumcycle - Lini)/Lini)*(Kcb_mid - Kcb_ini);
                     }
                     if(sumcycle >= Lini+Ldev && sumcycle < Lini+Ldev+Lmid) Kcb[ii] = Kcb_mid;
//(50-79)
                     if(sumcycle >= Lini+Ldev+Lmid  && sumcycle < Calendar){ //new (80-99)
                              Kcb[ii] = Kcb_mid + (( sumcycle -
(Lini+Ldev+Lmid))/Llate)*(Kcb_end - Kcb_mid);
                     }
                     if(sumcycle < Lini+Ldev){          //(0-49)
                            RD[jj][ii] = MinRD + (MaxRD - MinRD)*(Kcb[ii] - Kcb_ini)/(Kcb_mid -
Kcb_ini);
                            if(RD[jj][ii] > MaxRD) RD[jj][ii] = MaxRD; // new
                     }
                     else{
                            RD[jj][ii] = MaxRD;
                     }
                   }
                   else if(smc_rz[jj][ii-1] < WP_RZ[jj][ii-1]) {
                     if(sumcycle < Lini) Kcb[ii] = Ks[jj][ii]*Kcb_ini;
                     if(sumcycle >= Lini && sumcycle < Lini+Ldev ){
```

```
                        Kcb[ii] = Ks[jj][ii]*Kcb_ini + ((sumcycle -
Lini)/Lini)*(Ks[jj][ii]*Kcb_mid - Ks[jj][ii]*Kcb_ini);
                }
                if(sumcycle >= Lini+Ldev && sumcycle < Lini+Ldev+Lmid) Kcb[ii] =
Ks[jj][ii]*Kcb_mid;
                if(sumcycle >= Lini+Ldev+Lmid  && sumcycle < Calendar){ //new
                        Kcb[ii] = Ks[jj][ii]*Kcb_mid + (( sumcycle -
(Lini+Ldev+Lmid))/Llate)*(Ks[jj][ii]*Kcb_end - Ks[jj][ii]*Kcb_mid);
                }
                if(sumcycle < Lini+Ldev){
                        RD[jj][ii] = MinRD + (MaxRD - MinRD)*(Kcb[ii] -
Ks[jj][ii]*Kcb_ini)/(Ks[jj][ii]*Kcb_mid - Ks[jj][ii]*Kcb_ini);
                        if(RD[jj][ii] > MaxRD) RD[jj][ii] = MaxRD;
                }
                else{
                        RD[jj][ii] = MaxRD;
                }
                }//end of Kcb and RD adjust

                xx = fxloss[jj][ii];
                if(ii == 0){
                        if(IFLAG[jj][ii] == 1) {
                                irrig[jj][ii] = (float)(sup_wrbd[jj] - fxloss[jj][ii]) * IFLAG[jj][ii];
                                acumloss[ii] = 0.0f;
                        }
                        if(IFLAG[jj][ii] == 0) irrig[jj][ii] = 0.0f;
                }
                if(ii > 0) {
                        if(IFLAG[jj][ii-1] == 0 && IFLAG[jj][ii] == 1){
                                irrig[jj][ii]  = (float)(sup_wrbd[jj] - fxloss[jj][ii]) * IFLAG[jj][ii];
                                acumloss[ii]  = 0.0f;
                        }
                        if(IFLAG[jj][ii-1] == 1 && IFLAG[jj][ii] == 1){
                                if(irrig[jj][ii-1] == 0.0f){
                                        irrig[jj][ii]  = (float)(sup_wrbd[jj] - acumloss[ii-1])  *
IFLAG[jj][ii];
                                        xx = acumloss[ii-1];
                                }
                                else {
                                        irrig[jj][ii]  = sup_wrbd[jj] * IFLAG[jj][ii];
                                }
                        }
                        if(IFLAG[jj][ii] == 0) irrig[jj][ii] = 0.0f;
                }
                if(irrig[jj][ii] < 0.0f) {
                        lwirri[jj][ii] = (-1)*irrig[jj][ii];

                        irrig[jj][ii] = 0.0f;
                        acumloss[ii] = xx - sup_wrbd[jj];
                }
                else {
                        acumloss[ii] = 0.0;
                        lwirri[jj][ii] = 0.0f;
                }
                //end of new fixloss adjust
        }// end of ii loop preparation of all input data
// Now start of Main loop in every cycle
        for(ii = 0; ii <cycle[i][jj]; ii++){// Main LOOP for ii
```

```
                    // Start to strip soil layer in calculation here
                    nlr_RZ_MAX              = (int)(MaxRD/layerD); //no. of layers at RD=Max
                    NLR_RZ[jj][ii]  = (int)(RD[jj][ii]/layerD); // no. of layers at a time period
                    RD[jj][ii]              = (float)(NLR_RZ[jj][ii]*layerD); //adjust RD by layer
                    FC_RZ[jj][ii]     = FC[jj]*RD[jj][ii]*1000;
                    PWP_RZ[jj][ii]   = PWP[jj]*RD[jj][ii]*1000;
                    RAM_RZ[jj][ii]  = (FC[jj] - PWP[jj])*(RD[jj][ii]*1000)*P[ii];
                    WP_RZ[jj][ii]     = FC_RZ[jj][ii] - RAM_RZ[jj][ii]; //new for varying p
                    if(ii > t_lag[jj]){
                              if(NLR_RZ[jj][ii] > NLR_RZ[jj][ii-1]){
                                        tsum[jj][ii] = 0.0f;
                                        for(nsl = NLR_RZ[jj][ii-1]; nsl < NLR_RZ[jj][ii]; nsl++){
                                                  tsum[jj][ii] = tsum[jj][ii] + smc_layer[nsl][jj][ii-1];

                                        }
                                        smc_rz[jj][ii] = smc_rz[jj][ii-1] + tsum[jj][ii];
                              }
                              else smc_rz[jj][ii] = smc_rz[jj][ii-1];
                    }
// This is to sum up smc for global check
     if(ii == t_lag[jj]) sum_up_smc[jj][ii] = WP_RZ[jj][ii];
          else sum_up_smc[jj][ii] = smc_rz[jj][ii];
                    // Now calculate supply[jj][ii] in each time period with IFLAG[jj][ii]
     supply[jj][ii] =  irrig[jj][ii]; //irrig is amount of water supply per time period
                    sumsupply[i]   =  sumsupply[i] + supply[jj][ii] * scheme_area[jj] / tot_area;
//Soil Water Balance Main Equation
                    if(ii < t_lag[jj]){
                              Ea[jj][ii] = 0.0f;
                              SMCline[jj][ii] = 0.0f;
                    }
                    else if(ii==t_lag[jj]) {
                              Ea[jj][ii] = ETc[jj][ii]; //because initial smc = WP_RZ
                    smc_rz[jj][ii]  = (float)(WP_RZ[jj][ii] + supply[jj][ii] - Ea[jj][ii] + ER[jj][ii]);

                              if(smc_rz[jj][ii] < WP_RZ[jj][ii]){
                                        Ea[jj][ii] = (smc_rz[jj][ii] -
PWP_RZ[jj][ii])*(ETc[jj][ii])/(WP_RZ[jj][ii]-PWP_RZ[jj][ii]);
                                        smc_rz[jj][ii] = (float)(WP_RZ[jj][ii] + supply[jj][ii] - Ea[jj][ii] +
ER[jj][ii]);
                              }
                    }
                    else if(ii > t_lag[jj]){
                              if(smc_rz[jj][ii] >= WP_RZ[jj][ii]) {
                                        Ea[jj][ii] = ETc[jj][ii];
                                        smc_rz[jj][ii] = (float)(smc_rz[jj][ii] + supply[jj][ii] - Ea[jj][ii] +
ER[jj][ii]);
                              }
                              else if(smc_rz[jj][ii] < WP_RZ[jj][ii]){
                                        Ea[jj][ii] = (smc_rz[jj][ii] -
PWP_RZ[jj][ii])*(ETc[jj][ii])/(WP_RZ[jj][ii]-PWP_RZ[jj][ii]);
                                        smc_rz[jj][ii] = (float)(smc_rz[jj][ii] + supply[jj][ii] - Ea[jj][ii] +
ER[jj][ii]);
                              }
                    }
                    // end Soil Moisture Balance Main Equation and its adjustment.
                    SMCline[jj][ii] = smc_rz[jj][ii]; //for ploting
// this is when SMC > FC
                    infilt = 0.0f;
```

```
                if(smc_rz[jj][ii] > FC_RZ[jj][ii]){ //when smc_rz OVER FC_RZ > infil cal. here
                        SMCline[jj][ii] =  FC_RZ[jj][ii];//for plotting if smc_rz just above but not
oversupply
                        infilt = smc_rz[jj][ii] - FC_RZ[jj][ii];
                        infiltopup[jj][ii] = infilt; //this is to find how much infiltration top up in this
ii
                        smc_rz[jj][ii] =  FC_RZ[jj][ii];
                        // this is to put infiltration from layer to layer below
                        for(nsl = NLR_RZ[jj][ii]; nsl < nlr_RZ_MAX; nsl++) {
                                smc_layer[nsl][jj][ii] = smc_layer[nsl][jj][ii-1] + infilt;
                                if(smc_layer[nsl][jj][ii] > FC_layer){
                                        infilt = smc_layer[nsl][jj][ii] - FC_layer;
                                        smc_layer[nsl][jj][ii] = FC_layer;
                                }
                                else {
                                        infilt = 0.0;
                                        break;
                                }
                        }
                        Ovr_supl[jj][ii] = infilt;
                }
                else{
                        for(nsl = 0; nsl < nlr_RZ_MAX; nsl++) {
                        // if there is no infil so, smc_layer[ii] = smc_layer[ii-1] last cycle
                        smc_layer[nsl][jj][ii] = smc_layer[nsl][jj][ii-1];
                        }
                        infiltopup[jj][ii] = 0.0f;
                        Ovr_supl[jj][ii]  = 0.0f;
                }
                sumOversupl[i] = sumOversupl[i] + Ovr_supl[jj][ii];
// This is to compute net_infil this cycle
                if(smc_rz[jj][ii] > FC_RZ[jj][ii])     netinfil[jj][ii] = infiltopup[jj][ii] -
Ovr_supl[jj][ii];
                else netinfil[jj][ii] = 0.0f;
// This is for plotting smc line if oversupply
                if(Ovr_supl[jj][ii] > 0.0f) SMCline[jj][ii]  = FC_RZ[jj][ii] + Ovr_supl[jj][ii]; //new

// This is for calculation of Ks when smc_rz < WP_RZ
                if(smc_rz[jj][ii] < WP_RZ[jj][ii]) {
                        Ks[jj][ii] = (FC_RZ[jj][ii] - smc_rz[jj][ii])/(FC_RZ[jj][ii]-WP_RZ[jj][ii]);
                }
                else Ks[jj][ii] = 1;
// This is a penalty function.
                if(smc_rz[jj][ii] < WP_RZ[jj][ii]) {
                // Now must define once again incase if smc < pwp
                        LWP[jj][ii] = (float)(WP_RZ[jj][ii] - smc_rz[jj][ii]);//this if for scarcity
                        Equity[jj][ii] = (float)pow((R1*LWP[jj][ii]),2);
                // This is for plotting of SMCline if smc < WP
                SMCline[jj][ii] = WP_RZ[jj][ii] - LWP[jj][ii];
                }
                else {
                        LWP[jj][ii] = 0.0f;
                        Equity[jj][ii] = 0.0f;
                }
                sumLWP[i]    = sumLWP[i] + LWP[jj][ii];
    sumEquity[i] = sumEquity[i] + Equity[jj][ii];

//This is for SMC_line plotting
```

```
                    if(ii < t_lag[jj]) SMCline[jj][ii] = 0.0f;  //new
                    if(ii > t_lag[jj] + Calendar) IFLAG[jj][ii] = 0;

// this is for calculation of global SMC to recheck
                    sumsmc_rz[i][jj]  = sumsmc_rz[i][jj] + sum_up_smc[jj][ii];//this is for global check
                    sumX[i][jj]       = sumX[i][jj]      + supply[jj][ii];//excluded fixloss
                    sumETc[i][jj]     = sumETc[i][jj]    + ETc[jj][ii];
                    sumEa[i][jj]          = sumEa[i][jj]     + Ea[jj][ii];
                    sumSMCend[i][jj]  = sumSMCend[i][jj] + smc_rz[jj][ii];
                    sumO[i][jj]                    = sumO[i][jj]        + Ovr_supl[jj][ii]; //sum oversupply
above RDmax
                    cum_stress[i][jj] = cum_stress[i][jj]+ LWP[jj][ii];
                    sumER[i][jj]          = sumER[i][jj]    + ER[jj][ii];
                    suminfil[i][jj]   = suminfil[i][jj]  + infiltopup[jj][ii];
                    sumnetinfil[i][jj]= sumnetinfil[i][jj] + netinfil[jj][ii];
                    sumLwirr[i][jj]   = sumLwirr[i][jj]   + lwirri[jj][ii];
                    GlobalSMC[i][jj]  = sumsmc_rz[i][jj] + sumX[i][jj] - sumEa[i][jj] + sumER[i][jj] -
sumSMCend[i][jj] - suminfil[i][jj];
                    sumGlobal[i]      = sumGlobal[i] + GlobalSMC[i][jj];
                sumIrr_day[i][jj] = sumIrr_day[i][jj] + IFLAG[jj][ii];
            }//end of ii loop
}//end of jj scheme

// this is canal capacity constraint
  for(ii=0; ii< CYCLE; ii++){
            sumBflow[ii] = 0.0f;
     for(jj = 0; jj <NSCHEME; jj++) {
                    branchGO[ii][jj]= Bflowrate[jj]*IFLAG[jj][ii];
                    sumBflow[ii]     = sumBflow[ii] + branchGO[ii][jj];
            }
     if(sumBflow[ii] > (main_capa*PercentFlow/100)) {
                    capaConstr[i][ii] = (float)(R2*(sumBflow[ii] - main_capa*PercentFlow/100));
            }
     else capaConstr[i][ii] = 0.0f;
            sumCapaConstr[i] = sumCapaConstr[i] + (float)(pow(capaConstr[i][ii],2));
  }//end of ii in capacity constraint loop
// Objective function is Here
     ftnss[i] = (float)(sumsupply[i] + sumEquity[i] + sumCapaConstr[i]);
     sumftnss = sumftnss + ftnss[i];
             if(bestfit  > ftnss[i]){
                            bestfit  = ftnss[i];

                            psumLWP  = sumLWP[i];
          psumOVR  = sumOversupl[i];
          psumCapa  = sumCapaConstr[i];
                            psumsupply= sumsupply[i];
                            psumEquity= sumEquity[i];
                            for(jj = 0; jj<NSCHEME; jj++) {
                              psumX[jj] = sumX[i][jj];
                    psumEa[jj] = sumEa[i][jj];
                              psumETc[jj] = sumETc[i][jj];
          psumsmc_rz[jj] = sumsmc_rz[i][jj];
                            psumSMCend[jj] = sumSMCend[i][jj];
                                    psumER[jj] = sumER[i][jj];
                                     psumO[jj] = sumO[i][jj];
                                pstress[jj] = cum_stress[i][jj];
                               pGlobalSMC[jj] = GlobalSMC[i][jj];
                      psumIrr_day[jj] = sumIrr_day[i][jj];
```

```
            psuminfil[jj] = suminfil[i][jj];
                        psumnetinfil[jj] = sumnetinfil[i][jj];
                                    psumLwirr[jj] = sumLwirr[i][jj];
                                                pcycle[jj] = cycle[i][jj];
                                                 pstrt[jj] = strt[jj];
                  for(ii=0; ii<cycle[i][jj]; ii++) {
                       pFlag[jj][ii] =  IFLAG[jj][ii];
                                                      pFC_RZ[jj][ii] =  FC_RZ[jj][ii];
                                                    prnWP_RZ[jj][ii] =  WP_RZ[jj][ii];
                                                   prnPWP_RZ[jj][ii] =  PWP_RZ[jj][ii];
                                                            pRD[jj][ii] =  RD[jj][ii];
                                                            pEa[jj][ii] =  Ea[jj][ii];
                                                           pETc[jj][ii] =  ETc[jj][ii];
                                                          ptsum[jj][ii] =  tsum[jj][ii];
                                               pinfiltopup[jj][ii] =  infiltopup[jj][ii];
                  .       psupply[jj][ii] =  supply[jj][ii];
                                                        pdemand[jj][ii] =  demand[jj][ii];
                                               psmc_rz[jj][ii] =  smc_rz[jj][ii];
                                                        pSMCline[jj][ii] =  SMCline[jj][ii];
                       pOvr_supl[jj][ii] =  Ovr_supl[jj][ii];
                                                              pLWP[jj][ii] =  LWP[jj][ii];
                                                     pcapaConstr[ii] =  capaConstr[i][ii];

                                               }
                                    }
                        }
}//end of i

 if (gen == MaxGEN-1 || DELTA_ftnss <= DELTA ) {
            index = 999;
      for(jj = 0; jj<NSCHEME; jj++) {
                  N_fixloss         = (float)(psumNfix[jj]/fixloss[jj]);
                  fprintf(f1,"\n");
            fprintf(f1,"IFLAG    ");
       for(ii=0; ii<pcycle[jj]; ii++)  fprintf(f1,"%8d",pFlag[jj][ii]); fprintf(f1,"\n");
//                fprintf(f1,"RD       ");
//     for(ii=0; ii<pcycle[jj]; ii++)  fprintf(f1,"%8.3f",pRD[jj][ii]); fprintf(f1,"\n");
          fprintf(f1,"FC       ");
       for(ii=0; ii<pcycle[jj]; ii++)  fprintf(f1,"%8.3f",pFC_RZ[jj][ii]); fprintf(f1,"\n");
                  fprintf(f1,"tsum     ");
                  for(ii=0; ii<pcycle[jj]; ii++)  fprintf(f1,"%8.3f",ptsum[jj][ii]); fprintf(f1,"\n");
                  fprintf(f1,"supply   ");
                  for(ii=0; ii<pcycle[jj]; ii++)  fprintf(f1,"%8.3f",psupply[jj][ii]); fprintf(f1,"\n");
                  fprintf(f1,"ETc      ");
       for(ii=0; ii<pcycle[jj]; ii++)  fprintf(f1,"%8.3f",pETc[jj][ii]); fprintf(f1,"\n");
                  fprintf(f1,"Ea       ");
       for(ii=0; ii<pcycle[jj]; ii++)  fprintf(f1,"%8.3f",pEa[jj][ii]); fprintf(f1,"\n");
                  fprintf(f1,"smc_rz   ");
          for(ii=0; ii<pcycle[jj]; ii++)  fprintf(f1,"%8.3f",psmc_rz[jj][ii]); fprintf(f1,"\n");
          fprintf(f1,"SMC_line ");
       for(ii=0; ii<pcycle[jj]; ii++)  fprintf(f1,"%8.3f",pSMCline[jj][ii]);fprintf(f1,"\n");
       fprintf(f1,"WP       ");
       for(ii=0; ii<pcycle[jj]; ii++)  fprintf(f1,"%8.3f",prnWP_RZ[jj][ii]);          fprintf(f1,"\n");
       fprintf(f1,"PWP      ");
       for(ii=0; ii<pcycle[jj]; ii++)  fprintf(f1,"%8.3f",prnPWP_RZ[jj][ii]); fprintf(f1,"\n");
                  fprintf(f1,"BelowWP  ");
       for(ii=0; ii<pcycle[jj]; ii++)  fprintf(f1,"%8.3f",pLWP[jj][ii]); fprintf(f1,"\n");
```

```
                pEaETcRatio[jj] = psumEa[jj]/psumETc[jj];

                fprintf(f1,"Scheme  %4d\n",  jj+1);
                fprintf(f1,"Irrifrq %4.1f\n", N_fixloss);
                fprintf(f1,"Irr_day %4.1f\n", psumIrr_day[jj]);
                fprintf(f1,"Fixloss %4.1f\n", psumNfix[jj]);
                fprintf(f1,"strtTP  %4d\n",  pstrt[jj]);
      fprintf(f1,"Global %9.3f\n", pGlobalSMC[jj]);
                fprintf(f1,"SMCstr %9.3f\n", psumsmc_rz[jj]);
                fprintf(f1,"Irriga %9.3f\n", psumX[jj]);
                fprintf(f1,"ETc    %9.3f\n", psumETc[jj]);
                fprintf(f1,"Ea__   %9.3f\n", psumEa[jj]);
                fprintf(f1,"Ea/ETc %9.3f\n", pEaETcRatio[jj]);
                fprintf(f1,"ER     %9.3f\n", psumER[jj]);
                fprintf(f1,"SMCend %9.3f\n", psumSMCend[jj]);
                fprintf(f1,"Ovrsup %9.3f\n", psumO[jj]);
                fprintf(f1,"Lwrirr %9.3f\n", psumLwirr[jj]);
                fprintf(f1,"infilt %9.3f\n", psuminfil[jj]);
                fprintf(f1,"stress %9.3f\n", pstress[jj]);

          }// end of scheme
          fprintf(f1,"\n");
    fprintf(f1," %1.0f  %4.2f  %4.3f  %3d  %3d  ", Tperiod, PXOVER, NofMu, gen, PercentFlow);
    fprintf(f1," %7.3f  %7.3f  %2d  %7.3f  %2d  %7.3f\n",
                        bestfit, psumsupply, R1 , psumEquity, R2,  psumCapa);
  }// end of if loop
  avgftnss = sumftnss/POPSIZE;
  minftnss  = ftnss[0];
  for(i=0; i<POPSIZE; i++) {
    if(ftnss[i]  < minftnss)   minftnss  = ftnss[i];
          if(ftnss[i]  < ftnss[POPSIZE]){
                ftnss[POPSIZE] = ftnss[i];
          }
  }
}
/************************************/
void Tournament() {
        int  ii,jj,j,mem,ss;
        for(mem=0; mem < POPSIZE; mem++){
                ii = RandInt(0,POPSIZE-1);
                jj = RandInt(0,POPSIZE-1);
                if(ftnss[ii] <= ftnss[jj]){
                        for(ss =0; ss<NSCHEME; ss++){
                                for(j=0; j<LENGTH; j++){
                                        mated_pop[mem][ss][j] = initial_pop[ii][ss][j];
                                }
                        }
                }
     if(ftnss[ii] > ftnss[jj]){
                        for(ss =0; ss<NSCHEME; ss++){
                                for(j=0; j<LENGTH; j++){
                                        mated_pop[mem][ss][j] = initial_pop[jj][ss][j];
                                }
                        }
                }
        }
}
/************************************/
```

```
// Routine to crossover population
void XOVER(int i1, int i2) {
        int j,kk,ss;
        kk = RandInt(1,LENGTH-1);
        for(ss =0; ss<NSCHEME; ss++){
                for(j=kk;j<LENGTH;j++){
//                SwapIntValues(&mated_pop[i1][ss][j],&mated_pop[i2][ss][j]);
                SwapRealValues(&mated_pop[i1][ss][j],&mated_pop[i2][ss][j]);
                }
  }
}
/*************************************/
void UNIXOVER(int i1, int i2){
        int j,kk,ss;
        for(ss =0; ss<NSCHEME; ss++){
                for(j=0;j<LENGTH;j++)  {
                kk= rand()&01;
//      if(kk==1) SwapIntValues(&mated_pop[i1][ss][j],&mated_pop[i2][ss][j]);
                if(kk==1) SwapRealValues(&mated_pop[i1][ss][j],&mated_pop[i2][ss][j]);
                }
        }
}
/*************************************/
// Routine to crossover population, 2-site
void TPXOVER(int i1, int i2){
        int         j,kk1,kk2,ss;
        kk1 = RandInt(1,LENGTH-1);
        kk2 = RandInt(1,LENGTH-1); // this is for 2-site XOVER
        if (kk1>kk2) SwapIntValues(&kk1,&kk2); // this is to swap 2-site if the later is less
        for(ss =0; ss<NSCHEME; ss++){
                for(j=kk1;j<kk2;j++){
//      SwapIntValues(&mated_pop[i1][ss][j],&mated_pop[i2][ss][j]);
                SwapRealValues(&mated_pop[i1][ss][j],&mated_pop[i2][ss][j]);
                }
        }
}
/*************************************/
void SelectMate() {
        int  i,j,mem, num_select, one, ss;
  float Prob_X;
  num_select = 0;
        for (mem = 0; mem< POPSIZE ; mem++) {
                Prob_X = rand()%1000/1000.0f;
                if (Prob_X < PXOVER){
                        ++num_select;
                        if(num_select%2 == 0){
                                TPXOVER(one,mem);  // for testing 1-site use XOVER, for 2-site
use TPXOVER
                        }
                        else one = mem;
                }
        }
}
/*****************************/
void UniSelectMate() {
        int  i, j, mem, num_select,one,ss;
  float Prob_X;
```

```
  num_select = 0;
        for (mem = 0; mem< POPSIZE ; mem++) {
                Prob_X = rand()%1000/1000.0f;
                if (Prob_X < PXOVER){
                        ++num_select;
                        if(num_select%2 == 0){
                                UNIXOVER(one,mem);
                        }
                        else one = mem;
                }
        }
}
/*****************************************/
// this is to radom probability of mutation
void mutatepop() {
 int i,j,ss, mutant;
 double r;
          for (i=0;i<POPSIZE;i++) {
                for(ss =0; ss<NSCHEME; ss++){
                                r = rand()%1000/1000.0;
                                if(r<Pmutate){
                                        mutant = rand()&01;
                                        if(mutant == 1) mated_pop[i][ss][0] =
(float)(mated_pop[i][ss][0] + inc);
                                        else mated_pop[i][ss][0] = (float)(mated_pop[i][ss][0] -
inc);
                                }
                                if(mated_pop[i][ss][0] < MinLag) mated_pop[i][ss][0] =
(float)(MinLag);
                                else if(mated_pop[i][ss][0] > MaxLag-1) mated_pop[i][ss][0] =
(float)(MaxLag-1);
                           for(j=1;j<LENGTH;j++) {
           r = rand()%1000/1000.0;
                            if(r<Pmutate){
                                if(mated_pop[i][ss][j] == 0)  mated_pop[i][ss][j] = 1;
                                else mated_pop[i][ss][j] = 0;
                             }
                           }
                }
                for(ss =0; ss<NSCHEME; ss++){
                  for(j=0;j<LENGTH;j++)  {
        initial_pop[i][ss][j] = mated_pop[i][ss][j];
                  }
                }
          }
}
/*************************************/
// Routine to print error message
void nrerror(char *messg){
        puts(messg) ;
   exit(1) ;
}
/*************************************/
// Routine to swop value
void SwapIntValues(int *x, int *y){
        int temp;
        temp = *x;
        *x=*y;
```

```
        *y=temp;
}
/*************************************/
// Routine to swop real value
void SwapRealValues(float *xx, float*yy) {
        float temp;
        temp = *xx;
        *xx = *yy;
        *yy = temp;
}
/*************************************/
```

```
//GA47W.c      Program to formulate GAs for water scheduling problem
//             WARABANDI Approach for Chapter 8
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <process.h>
#include <time.h>

#define Tperiod      1.0     // input time period here (0.5 or 1.0)
#define POPSIZE      100     // input population size
#define INTERVAL     300     // input interval of generation to recheck of improvement fitness
#define DELTA        0.0001  // input minimum improvement of fitness after INTERVAL
                             generations
#define NUMCROP      1       //input number of crop in this system
#define NSCHEME      15      // input total number of schemes
#define Max_wrbd     14      // input allowable maximum warabandi
#define Min_wrbd     7       // input allowable minimum warabandi
#define MaxLag       15      // input maximum stagerring days from the start
#define MinLag       0       // input minimum staggering days
#define Calendar     195     // input total growth periods
#define STAGE        195     // input stage of growth define related to crop requirement data file
#define CYCLE        210     //(Calendar/Tperiod)+ max. lag days
#define LENGTH       3       // input chromosome length required for each tertiary
#define inc          Tperiod // increment of 1.0 day to mutate wara and shutdown (day)
#define inc2         7       // input increment of warabandi (e.g. 7)
#define IrriEff      0.9     // input irrigation efficiency
#define  layerD      0.01    // input a root zone soil layer for water balance in meter
#define  tDepth      1.5     // input total depth of soil in meter
#define MaxLyr       150     // input max. number of soil layers

int     RandInt (int, int);
float   RandReal (float,float);
void    Read_inputfiles();
void    SwapIntValues(int *, int *);
void    SwapRealValues(float *, float *);
void    nrerror(char *messg) ;
void    initial();
void    objective();
void    SelectMate();
void    UniSelectMate();
void    mutatepop();
void    Tournament();

int     MaxGEN, R1, R2, PercentFlow;
float   t_lag, NofMu, PXOVER;
int     g, gen, i ,tt, index, stage_day[STAGE],stagenum[STAGE];
float   mated[POPSIZE],bestfit, Pmutate, ftnss[POPSIZE+1], DELTA_ftnss, OLD_minftnss;
float   initial_pop[POPSIZE][NSCHEME][LENGTH],
        mated_pop[POPSIZE][NSCHEME][LENGTH];
float   MinRD, MaxRD, Lini, Ldev, Lmid, Llate, Kcb_ini, Kcb_mid, Kcb_end, Kcb[CYCLE];
float   D_factr, FC[NSCHEME], PWP[NSCHEME];
float   Bflowrate[NSCHEME], input_ETc[MaxLag][STAGE], main_capa;
float   scheme_area[NSCHEME], Eff_rain[MaxLag][STAGE];
float   tot_area, fixloss[NSCHEME],     fxloss[NSCHEME][CYCLE];
float   strt[NSCHEME], pstrt[NSCHEME];
float   wara[NSCHEME], pwara[NSCHEME];
```

```
float  shut[NSCHEME], pshut[NSCHEME];

FILE *f1,*f2,*f3 ;
//MAIN PROGRAM
void main (void){
   clock_t start, finish;
   float  duration, TotalTime ;
   TotalTime = 0.0f;
   if(!(f1 = fopen("ga47Winc.out","w"))) {
                  printf("can't open output file\n");
                  exit(1);
         }
   Read_inputfiles();
         srand(1);
   start = clock();
   MaxGEN      =        2500;
         PXOVER      =        0.9f;
         NofMu  =        0.3f;
         R1          =        1;
         R2          =        12500;
         PercentFlow = 100;
//       R1    =  1;
//do{
//       R2          =        5000;
//do{
//       NofMu  =        0.1f;
//do{
//       PXOVER      =        0.5f;
//do{
//       PercentFlow = 85;
//do{
   initial();
   index = 0;
         OLD_minftnss    = 10000.0f;
         DELTA_ftnss           = 10000.0f;
         bestfit               = 1000000.0f;
         ftnss[POPSIZE]  = 1000000.0f;
 for(gen = 0; gen < MaxGEN; gen++) {
         objective(); if(index == 999) break;
   Tournament();
//       SelectMate();
         UniSelectMate();
   mutatepop();
   if (!((gen+1)%INTERVAL)){
     DELTA_ftnss = (float)fabs((double)((OLD_minftnss - ftnss[POPSIZE])/ftnss[POPSIZE]));
     OLD_minftnss = ftnss[POPSIZE];
   }
 } //End of Generation Loop
// PercentFlow = PercentFlow + 5;
//}while (PercentFlow <= 91);
// PXOVER = (float) (PXOVER + 0.05f);
//       }while (PXOVER <= 0.71);
//       NofMu = (float)(NofMu + 0.1f);
//       }while (NofMu <= 0.31);
//       R2 = R2 + 2500;
//       }while (R2 <= 12501);
//       R1 = R1 + 4;
//       }while (R1 <= 6);
```

```
 finish = clock();
 duration = (float)(finish - start) / CLOCKS_PER_SEC;
 TotalTime = (float)(TotalTime + duration);
 printf(" Time = %5.2f minutes\n", duration/60);
 fprintf(f1,"Time = %5.2f minutes \n\n", duration/60);
 (void) fclose (f1);
 printf("End of run  ENJOY!!!! ");
} //MAIN PROGRAM END
void Read_inputfiles(){
  int  ii, jj, kk, num_schemes, scheme_node[NSCHEME];
       char  title[80],title2[80],title3[80],title4[80],title5[80],title6[80],title7[80],title8[80];
       char  title9[80], title10[80], title11[80], title12[80], schedulefile[20];
       if(!(f2 = fopen("Proj47W.lst","r"))) {  //for input47W.dat
   nrerror(" Can not open project file");
  }
  // now read the data file names
       fgets(schedulefile, 20, f2) ;
       for(ii = 0; ii < 20; ii++) {
            if(schedulefile[ii] == '\n') schedulefile[ii] = '\0' ;
       }
       (void) fclose(f2) ;
       if(!(f2 = fopen(schedulefile,"r"))) {
    nrerror(" Unable to open schedulefile") ;
  }
  fgets(title, 80, f2);          // Water scheduling title
  fscanf(f2,"%d",&num_schemes);  // scan number of schemes here
  for(jj=0;jj<num_schemes;jj++) fscanf(f2,"%d ", &scheme_node[jj]);
  fgets(title2, 80, f2);         // scheme area (ha) title
  for(jj=0;jj<num_schemes;jj++) fscanf(f2,"%f ", &scheme_area[jj]);
       tot_area = 0.0f;
       for(jj=0;jj<num_schemes;jj++){
            tot_area = tot_area + scheme_area[jj];
       }
  fgets(title3, 80, f2);         // main canal supply title
       fscanf(f2,"%f ", &main_capa) ; // scan main canal supply
  fgets(title4, 80, f2);         // branch canal title
       for(jj = 0; jj < num_schemes; jj++) fscanf(f2,"%f ", &Bflowrate[jj]) ;
       fscanf(f2,"\n");
  fgets(title5, 80, f2);         // CWR title
  for(tt=0; tt<STAGE; tt++) fscanf(f2,"%d ", &stagenum[tt]);
  for(tt=0; tt<STAGE; tt++) fscanf(f2,"%d ", &stage_day[tt]);
  for(kk = 0; kk < MaxLag; kk++) {
            for(tt = 0; tt <STAGE; tt++){
              fscanf(f2,"%f", &input_ETc[kk][tt]) ;
            }
       }
  fscanf(f2,"\n");
  fgets(title6, 80, f2);         // ER title
       for(kk = 0; kk< MaxLag; kk++){
            for(tt = 0; tt <STAGE; tt++){
                 fscanf(f2,"%f", &Eff_rain[kk][tt]) ;
            }
       }
  fscanf(f2,"\n");
       fgets(title7, 80, f2);    // fix loss title
       for(jj = 0; jj < num_schemes; jj++) fscanf(f2,"%f ", &fixloss[jj]) ;
       fscanf(f2,"\n");
       for(ii=0; ii<CYCLE; ii++){
```

```
                for(jj = 0; jj < num_schemes; jj++){
                fxloss[jj][ii] = fixloss[jj];
                } 
        }
  fgets(title8, 80, f2);        // field capacity title
        for(jj = 0; jj < num_schemes; jj++) {
                fscanf(f2,"%f ", &FC[jj]) ;
        }
  fgets(title9, 80, f2);        // permanent wilting point title
        for(jj = 0; jj < num_schemes; jj++) {
                fscanf(f2,"%f ", &PWP[jj]) ;
        }
  fscanf(f2,"\n");
        fgets(title10,80,f2);   // Length stage title
        fscanf(f2,"%f %f %f %f",&Lini, &Ldev, &Lmid, &Llate);
  fscanf(f2,"\n");
        fgets(title11,80,f2);   // Kcb title
        fscanf(f2,"%f %f %f",&Kcb_ini, &Kcb_mid, &Kcb_end);
  fscanf(f2,"\n");
  fgets(title12, 80, f2); // rooting depth title
        fscanf(f2,"%f %f", &MinRD, &MaxRD);
  fscanf(f2,"\n");
        fscanf(f2,"%f",&D_factr);
  (void)fclose(f2);
}//end Readinput file
/*********/
float RandReal (float lower,float upper){
        float val;
        float nb = ( upper - lower );
        val =  (float)(rand()%1000/1000.0*nb + lower);
        return ((float)val);
}
/*********/
int RandInt (int lower,int upper){
        int val;
        int nb = (upper - lower );
        val =  (rand()%nb + lower);
        return val;
}
/*********/
void initial(){
  int i, j;
  float nd = RAND_MAX;

  for(i = 0; i<POPSIZE; i++) {
          for(j = 0; j< NSCHEME; j++){
                  initial_pop[i][j][0] = (float)(MinLag   + inc*(float)floor((1+((MaxLag-1) -
MinLag)/inc)*(float)rand()/(nd+1.00)));
        initial_pop[i][j][1] = (float)RandInt(1,Max_wrbd);
        initial_pop[i][j][2] = (float)RandInt(1,Max_wrbd);
          }
  }
}
/************************/
void  objective(){
        int                 i, j, ii, cycle[POPSIZE][NSCHEME], countday, cvrs;
        int           IFLAG[NSCHEME][CYCLE],   pFlag[NSCHEME][CYCLE];
        float      sumftnss, minftnss, avgftnss;
```

```
        float    odds[POPSIZE][NSCHEME], odd2[NSCHEME][STAGE], sumcycle[NSCHEME];
        float    sumii, rota[NSCHEME];
        float    Ovr_supl[NSCHEME][CYCLE],pOvr_supl[NSCHEME][CYCLE];
        float    LWP[NSCHEME][CYCLE],              pLWP[NSCHEME][CYCLE];
        float    pETc[NSCHEME][CYCLE],      ETc[NSCHEME][CYCLE];
        float    sup_wrbd[NSCHEME];
        float    supply[NSCHEME][CYCLE], psupply[NSCHEME][CYCLE];
        float    demand[NSCHEME][CYCLE], pdemand[NSCHEME][CYCLE];
        float    psumOVR, psumLWP, RD[NSCHEME][CYCLE];
        float    sumLWP[POPSIZE];
        float    capaConstr[NSCHEME][CYCLE], pcapaConstr[NSCHEME][CYCLE];
        float  GlobalSMC[POPSIZE][NSCHEME], pGlobalSMC[NSCHEME];
        float    sumX[POPSIZE][NSCHEME], sumD[POPSIZE][NSCHEME];
        float  psumD[NSCHEME], psumX[NSCHEME];
        float    sumsmc_rz[POPSIZE][NSCHEME], psumsmc_rz[NSCHEME];
        float  smc_rz[NSCHEME][CYCLE], psmc_rz[NSCHEME][CYCLE];
        float  sumO[POPSIZE][NSCHEME], sumGlobal[POPSIZE], sumsupply[POPSIZE];
        float  MnCapaCon[CYCLE], sumMCapaCon[POPSIZE], psumMCapaCon;
        float  sumSecondaryFlow[CYCLE];
        float    SMCline[NSCHEME][CYCLE], pSMCline[NSCHEME][CYCLE];
        float    sumEquity[POPSIZE], psumsupply, psumEquity, Equity[NSCHEME][CYCLE];
        float    sumOversupl[POPSIZE];
        float    P[CYCLE], Ks[NSCHEME][CYCLE];
        int      NLR_RZ[NSCHEME][CYCLE], nlr_RZ_MAX, nsl, countn;
        float  FC_layer, RAM_layer, WP_layer, RAM_RZ[NSCHEME][CYCLE];
        float    FC_RZ[NSCHEME][CYCLE], PWP_RZ[NSCHEME][CYCLE],
WP_RZ[NSCHEME][CYCLE];
        float    smc_layer[150][NSCHEME][CYCLE], tsum[NSCHEME][CYCLE];
        float    pFC_RZ[NSCHEME][CYCLE], prnWP_RZ[NSCHEME][CYCLE];
        float    prnPWP_RZ[NSCHEME][CYCLE];
        float  Ea[NSCHEME][CYCLE], infilt ;
        float  pstress[NSCHEME], cum_stress[POPSIZE][NSCHEME];
        float  ER[NSCHEME][CYCLE], irrig[NSCHEME][CYCLE];
        float    sum_up_smc[NSCHEME][CYCLE];
        float    infiltopup[NSCHEME][CYCLE];
        float    sumEa[POPSIZE][NSCHEME];
        float    suminfil[POPSIZE][NSCHEME],sumSMCend[POPSIZE][NSCHEME];
        float    sumIrr_day[POPSIZE][NSCHEME], psuminfil[NSCHEME];
        float    psumEa[NSCHEME],psumSMCend[NSCHEME];
        float    psumO[NSCHEME];
        float    psumIrr_day[NSCHEME], pEa[NSCHEME][CYCLE] ;
        float    ptsum[NSCHEME][CYCLE], N_fixloss, B_GO[CYCLE][NSCHEME];
        float    sumETc[POPSIZE][NSCHEME],psumETc[NSCHEME];
        float    cnvrs_ETc[NSCHEME][STAGE], pEaETcRatio[NSCHEME];
        float    cnvrs_rain[NSCHEME][STAGE];
        int    pcycle[NSCHEME];
        float    sumCROP_DIE[POPSIZE][NSCHEME],CROP_DIE[NSCHEME][CYCLE];
        float    acumloss[CYCLE], xx, sumLwirr[POPSIZE][NSCHEME],
        lwirri[NSCHEME][CYCLE];
        float    sumER[POPSIZE][NSCHEME], psumER[NSCHEME];
        sumftnss = 0.0f;
for(i=0;i<POPSIZE;i++) {
        ftnss[i]          = 0.0f;
        sumLWP[i]                 = 0.0f;
        sumOversupl[i]  = 0.0f;
        sumMCapaCon[i]          = 0.0f;
        sumsupply[i]     = 0.0f;
        sumGlobal[i]     = 0.0f;
```

```
        sumEquity[i]      = 0.0f;
        // now transfer initial pop
        for(j = 0; j < NSCHEME; j++){
                        strt[j] = initial_pop[i][j][0];
                        wara[j] = initial_pop[i][j][1];
                        shut[j] = initial_pop[i][j][2];
                        t_lag = strt[j];
                        cycle[i][j] = (int)(Calendar+t_lag);
  // compute supply from this point
        sup_wrbd[j] =
(float)(IrriEff*Bflowrate[j]*Tperiod*3600*24*1000)/(scheme_area[j]*10000);
        odds[i][j]  = (float)(Calendar - (Tperiod*(int)floor(Calendar/Tperiod)) );
        Pmutate     = (float)(NofMu/(LENGTH*NSCHEME));
        // LENGTH = length of a secondary
        // a chromosome length is then equal to LENGTH*NSCHEME
        }
        // now transfer cnvrs_ETc by strt[j]
        for(j = 0; j <NSCHEME; j++) {
                cvrs = (int)(strt[j]);
                for(tt = 0; tt < STAGE; tt++){
                        cnvrs_ETc[j][tt]  = input_ETc[cvrs][tt];
                        cnvrs_rain[j][tt] = Eff_rain[cvrs][tt];
                }
        }
        // every stage of month it has odd2 that occours when devide stageday with Tperiod
        for(j = 0; j < NSCHEME; j++){ // LOOP J2
   for(tt = 0; tt < STAGE; tt++){
         if(tt == 0) {
                odd2[j][tt] = (float)(stage_day[tt] - ((int)floor(stage_day[tt]/Tperiod))*Tperiod );
         }
         else{
         odd2[j][tt] = (float)(odd2[j][tt-1] + stage_day[tt] - ((int)floor((odd2[j][tt-1] +
        stage_day[tt])/Tperiod))*Tperiod );
         }
         }
        } // end  J2
        // This is to transfer initial_pop to be IFLAG
        for(j = 0; j < NSCHEME; j++){
                rota[j] = wara[j] + shut[j];
                        t_lag   = strt[j];
                        sumii   =       0.0f;
                        for(ii      = 0; ii< CYCLE; ii++){
                                sumii++;
                                IFLAG[j][ii] = 0;
                                if(sumii >  t_lag && sumii <= wara[j]+t_lag) IFLAG[j][ii] = 1;
                                if(sumii >  wara[j]+t_lag) {
                                        if(sumii <= rota[j]+t_lag) IFLAG[j][ii] = 0;
                                        else {
                                                IFLAG[j][ii] = 1;
                                                sumii = t_lag+ 1.0f;
                                        }
                                }
                           if(ii >= t_lag+Calendar) IFLAG[j][ii] = 0;
                        }
        }       // end transfer array
//MAIN SCHEME LOOP for every branch OPEN HERE
        for(j = 0; j< NSCHEME; j++) {
     sumX[i][j]             = 0.0f;
```

```
                sumEa[i][j]    = 0.0f;
                sumETc[i][j]   = 0.0f;
                suminfil[i][j]  = 0.0f;
                sumO[i][j]     = 0.0f;
                sumsmc_rz[i][j] = 0.0f;
                cum_stress[i][j]  = 0.0f;
                sumIrr_day[i][j] = 0.0f;
                sumSMCend[i][j]  = 0.0f;
                sumLwirr[i][j]     = 0.0f;
                sumER[i][j]        = 0.0f;
    tt = 0;
    sumcycle[j] = 0;
          countday   =  stage_day[0];
    t_lag = strt[j];
        for(ii = 0; ii< cycle[i][j]; ii++){   // MAIN LOOP CYCLE is HERE
                if(ii < t_lag) {
                        ETc[j][ii] = 0.0f;
                        ER[j][ii] = 0.0f;
                }
    else if(ii >= t_lag){
                        sumcycle[j] = (float)(sumcycle[j] + Tperiod);
                if(sumcycle[j] <= countday){
                                ETc[j][ii] = (float)( (cnvrs_ETc[j][tt]/stage_day[tt])*Tperiod );
                                ER[j][ii] = (float)( (cnvrs_rain[j][tt]/stage_day[tt])*Tperiod );
                }
                else{
        if(stagenum[tt] != STAGE ){
          ETc[j][ii] = (float)(odd2[j][tt]*(cnvrs_ETc[j][tt])/stage_day[tt] + (Tperiod -
odd2[j][tt])*(cnvrs_ETc[j][tt+1])/stage_day[tt+1] );
          ER[j][ii] = (float)(odd2[j][tt]*(cnvrs_rain[j][tt])/stage_day[tt] + (Tperiod -
odd2[j][tt])*(cnvrs_rain[j][tt+1])/stage_day[tt+1] );
        }
         else {
        ETc[j][ii] = (float)(odd2[j][tt]*(cnvrs_ETc[j][tt])/stage_day[tt] );
        ER[j][ii] = (float)(odd2[j][tt]*(cnvrs_rain[j][tt])/stage_day[tt] );
          }
tt++;
        countday = countday + stage_day[tt];
        }
        }
// This is to adjust D_factor
                if(ETc[j][ii] > 5.0f){
                        P[ii] = (float)(D_factr + 0.04*(5 - ETc[j][ii]));
                        if(P[ii] < 0.1) P[ii] = 0.1f;
                        if(P[ii] > 0.8) P[ii] = 0.8f;
                }
                else P[ii] = D_factr;
//Set up smc in every soil strip layer as wilting point of every soil layer
                        FC_layer        = (float)(FC[j]*layerD*1000);
                        RAM_layer       = (float)((FC[j] - PWP[j])*layerD*1000*P[ii]);
                        WP_layer        = FC_layer - RAM_layer;
                        for(nsl = 0; nsl < MaxLyr; nsl++) {
                                smc_layer[nsl][j][ii-1] = WP_layer;
                        }
// for Kcb adjust and then cal of RD adjust
                if(ii < t_lag) RD[j][ii] = 0.0f;
                else if(ii >= t_lag || smc_rz[j][ii-1] >= WP_RZ[j][ii-1]) {
                  if(sumcycle[j] < Lini) Kcb[ii] = Kcb_ini;
```

```
                if(sumcycle[j] >= Lini && sumcycle[j] < Lini+Ldev ){
                        Kcb[ii] = Kcb_ini + ((sumcycle[j] - Lini)/Lini)*(Kcb_mid - Kcb_ini);
                }
        if(sumcycle[j] >= Lini+Ldev && sumcycle[j] < Lini+Ldev+Lmid) Kcb[ii] = Kcb_mid;
        if(sumcycle[j] >= Lini+Ldev+Lmid && sumcycle[j] < Calendar){
         Kcb[ii] = Kcb_mid + (( sumcycle[j] - (Lini+Ldev+Lmid))/Llate)*(Kcb_end - Kcb_mid);
        }
        if(sumcycle[j] < Lini+Ldev){
                RD[j][ii] = MinRD + (MaxRD - MinRD)*(Kcb[ii] - Kcb_ini)/(Kcb_mid - Kcb_ini);
                if(RD[j][ii] > MaxRD) RD[j][ii] = MaxRD;
        }
        else{
                RD[j][ii] = MaxRD;
        }
       }
       else if(smc_rz[j][ii-1] < WP_RZ[j][ii-1]) {
        if(sumcycle[j] < Lini) Kcb[ii] = Ks[j][ii]*Kcb_ini;
        if(sumcycle[j] >= Lini && sumcycle[j] < Lini+Ldev ){
       Kcb[ii] = Ks[j][ii]*Kcb_ini + ((sumcycle[j] - Lini)/Lini)*(Ks[j][ii]*Kcb_mid -
Ks[j][ii]*Kcb_ini);
        }
        if(sumcycle[j] >= Lini+Ldev && sumcycle[j] < Lini+Ldev+Lmid)
                Kcb[ii] = Ks[j][ii]*Kcb_mid;
                if(sumcycle[j] >= Lini+Ldev+Lmid && sumcycle[j] < Calendar){
                Kcb[ii] = Ks[j][ii]*Kcb_mid + (( sumcycle[j] -
(Lini+Ldev+Lmid))/Llate)*(Ks[j][ii]*Kcb_end - Ks[j][ii]*Kcb_mid);
                }
                if(sumcycle[j] < Lini+Ldev){
                RD[j][ii] = MinRD + (MaxRD - MinRD)*(Kcb[ii] -
                Ks[j][ii]*Kcb_ini)/(Ks[j][ii]*Kcb_mid - Ks[j][ii]*Kcb_ini);
                if(RD[j][ii] > MaxRD) RD[j][ii] = MaxRD;
                }
                else{
                        RD[j][ii] = MaxRD;
                 }
                }//end of Kcb and RD adjust
//for new fix loss adjust
                xx = fxloss[j][ii];
                if(ii == 0){
                        if(IFLAG[j][ii] == 1) {
                                irrig[j][ii] = (float)(sup_wrbd[j] - fxloss[j][ii]) * IFLAG[j][ii];
                                acumloss[ii] = 0.0f;
                        }
                        if(IFLAG[j][ii] == 0) irrig[j][ii] = 0.0f;
                }
                if(ii > 0) {
                        if(IFLAG[j][ii-1] == 0 && IFLAG[j][ii] == 1){
                                irrig[j][ii]  = (float)(sup_wrbd[j] - fxloss[j][ii]) * IFLAG[j][ii];
                                acumloss[ii]  = 0.0f;
                        }
                        if(IFLAG[j][ii-1] == 1 && IFLAG[j][ii] == 1){
                                if(irrig[j][ii-1] == 0.0f){
                                        irrig[j][ii]  = (float)(sup_wrbd[j] - acumloss[ii-1])  *
                                        IFLAG[j][ii];
                                        xx = acumloss[ii-1];
                                }
                                else {
                                        irrig[j][ii]  = sup_wrbd[j] * IFLAG[j][ii];
```

```
                                        }
                                }
                                if(IFLAG[j][ii] == 0) irrig[j][ii] = 0.0f;
                        }
                        if(irrig[j][ii] < 0.0f) {
                                lwirri[j][ii] = (-1)*irrig[j][ii];
                                irrig[j][ii] = 0.0f;
                                acumloss[ii] = xx - sup_wrbd[j];
                        }
                        else {
                                acumloss[ii] = 0.0;
                                lwirri[j][ii] = 0.0f;
                        }
                }// end of ii loop preparation of all input data

// Now start of Main loop in every cycle
                for(ii = 0; ii < cycle[i][j]; ii++) {// Main LOOP for ii
// start to strip soil layer in calculation here
                        nlr_RZ_MAX                      = (int)(MaxRD/layerD); //no. of layers at RD=Max
                        NLR_RZ[j][ii]   = (int)(RD[j][ii]/layerD); // no. of layers at a time period
                        RD[j][ii]       = (float)(NLR_RZ[j][ii]*layerD); //adjust RD by layer
                        FC_RZ[j][ii]        = FC[j]*RD[j][ii]*1000;
                        PWP_RZ[j][ii]   = PWP[j]*RD[j][ii]*1000;
                        RAM_RZ[j][ii]   = (FC[j] - PWP[j])*(RD[j][ii]*1000)*P[ii];
                        WP_RZ[j][ii]        = FC_RZ[j][ii] - RAM_RZ[j][ii];
                        if(ii > t_lag ){
                                if(NLR_RZ[j][ii] > NLR_RZ[j][ii-1]){
                                        tsum[j][ii] = 0.0f;
                                        for(nsl = NLR_RZ[j][ii-1]; nsl < NLR_RZ[j][ii]; nsl++){
                                                tsum[j][ii] = tsum[j][ii] + smc_layer[nsl][j][ii-1];
                                        }
                                        smc_rz[j][ii]   = smc_rz[j][ii-1] + tsum[j][ii];
                                }
                                else smc_rz[j][ii]  = smc_rz[j][ii-1];
                        }
// This is to sum up smc for global check
        if(ii == t_lag) sum_up_smc[j][ii] = WP_RZ[j][ii];
                else sum_up_smc[j][ii]  = smc_rz[j][ii];
//now calculate supply in each t-period
                supply[j][ii] = irrig[j][ii];
                        sumsupply[i]  = sumsupply[i] + supply[j][ii] * scheme_area[j] / tot_area;
//Soil Water Balance Main Equation
                        if(ii < t_lag) {
                                Ea[j][ii] = 0.0f;
                                SMCline[j][ii] = 0.0f;
                        }
                        else if(ii==t_lag) {
                                Ea[j][ii] = ETc[j][ii]; //because initial smc = WP_RZ
                        smc_rz[j][ii]  = (float)(WP_RZ[j][ii] + supply[j][ii] - Ea[j][ii] + ER[j][ii]);

                                if(smc_rz[j][ii] < WP_RZ[j][ii]){
                                        Ea[j][ii] = (smc_rz[j][ii] -
PWP_RZ[j][ii])*(ETc[j][ii])/(WP_RZ[j][ii]-PWP_RZ[j][ii]);
                                        smc_rz[j][ii] = (float)(WP_RZ[j][ii] + supply[j][ii] - Ea[j][ii] +
ER[j][ii]);
                                }
                        }
                        else if(ii > t_lag){
```

```
                        if(smc_rz[j][ii] >= WP_RZ[j][ii]) {
                                Ea[j][ii] = ETc[j][ii];
                                smc_rz[j][ii] = (float)(smc_rz[j][ii] + supply[j][ii] - Ea[j][ii] +
ER[j][ii]);
                        }
                        else if(smc_rz[j][ii] < WP_RZ[j][ii]){
                                Ea[j][ii] = (smc_rz[j][ii] -
PWP_RZ[j][ii])*(ETc[j][ii])/(WP_RZ[j][ii]-PWP_RZ[j][ii]);
                                smc_rz[j][ii] = (float)(smc_rz[j][ii] + supply[j][ii] - Ea[j][ii] +
ER[j][ii]);
                        }
                } // end Soil Moisture Balance Main Equation and its adjustment.
                SMCline[j][ii] = smc_rz[j][ii]; //for ploting
//then set smc_rz[jj][ii] = PWP_RZ[jj][ii]
                        smc_rz[j][ii]  = PWP_RZ[j][ii];
                }
// this is when SMC > FC
                infilt = 0.0f;
                if(smc_rz[j][ii] > FC_RZ[j][ii]){ // when smc_rz OVER FC_RZ > infil cal. here
                        SMCline[j][ii] = FC_RZ[j][ii];
                        infilt = smc_rz[j][ii] - FC_RZ[j][ii];
                        infiltopup[j][ii] =  infilt;
                        smc_rz[j][ii] =  FC_RZ[j][ii];
                        for(nsl = NLR_RZ[j][ii]; nsl < nlr_RZ_MAX; nsl++) {
                                smc_layer[nsl][j][ii] = smc_layer[nsl][j][ii-1] + infilt;
                                if(smc_layer[nsl][j][ii] > FC_layer){
                                        infilt = smc_layer[nsl][j][ii] - FC_layer;
                                        smc_layer[nsl][j][ii] = FC_layer;
                                }
                                else {
                                        infilt = 0.0;
                                        break;
                                }
                        }
                        Ovr_supl[j][ii] = infilt;
                }
                else{
                        for(nsl = 0; nsl < nlr_RZ_MAX; nsl++) {
                                smc_layer[nsl][j][ii] = smc_layer[nsl][j][ii-1];
                        }
                        infiltopup[j][ii] = 0.0f;
                        Ovr_supl[j][ii] = 0.0f;
                }
                sumOversupl[i] = sumOversupl[i] + Ovr_supl[j][ii];
// This is for plotting smc line if oversupply
                if(Ovr_supl[j][ii] > 0.0f) SMCline[j][ii]  = FC_RZ[j][ii] + Ovr_supl[j][ii];

// this is for calculation of Ks when smc_rz < WP_RZ
                if(smc_rz[j][ii] < WP_RZ[j][ii]) {
                        Ks[j][ii] = (FC_RZ[j][ii] - smc_rz[j][ii])/(FC_RZ[j][ii]-WP_RZ[j][ii]);
                }
                else Ks[j][ii] = 1;
// this is a penalty function
                if(smc_rz[j][ii] < WP_RZ[j][ii]) {
                        LWP[j][ii] = (float)(WP_RZ[j][ii] - smc_rz[j][ii]);
                        Equity[j][ii] = (float)pow((R1*LWP[j][ii]),2);
//this is for plotting
                        SMCline[j][ii] = WP_RZ[j][ii] - LWP[j][ii];
```

```
                    }
                    else {
                            LWP[j][ii] = 0.0f;
                            Equity[j][ii] = 0.0f;
                    }
                    sumLWP[i] = sumLWP[i] + LWP[j][ii];
      sumEquity[i] = sumEquity[i] + Equity[j][ii];
                    if(ii < t_lag) SMCline[j][ii] = 0.0f; // for plotting
// This is to compute Global Balance Check
                    sumsmc_rz[i][j]   = sumsmc_rz[i][j] + sum_up_smc[j][ii];
                    sumX[i][j]                = sumX[i][j]     + supply[j][ii];
                    sumETc[i][j]      = sumETc[i][j]   + ETc[j][ii];
                    sumEa[i][j]         = sumEa[i][j]    + Ea[j][ii];
                    sumSMCend[i][j]  = sumSMCend[i][j] + smc_rz[j][ii];
             sumO[i][j]     = sumO[i][j]     + Ovr_supl[j][ii];
                    cum_stress[i][j] = cum_stress[i][j]+ LWP[j][ii];
                    suminfil[i][j]   = suminfil[i][j]  + infiltopup[j][ii];
                    sumER[i][j]               = sumER[i][j]    + ER[j][ii];
                    sumLwirr[i][j]   = sumLwirr[i][j] + lwirri[j][ii];
                    GlobalSMC[i][j]  = sumsmc_rz[i][j] + sumX[i][j] - sumEa[i][j] + sumER[i][j] -
suminfil[i][j] - sumSMCend[i][j];
                    sumIrr_day[i][j] = sumIrr_day[i][j] + IFLAG[j][ii];
                    }// end of ii Main loop
}// end of j main loop
//This is for capacity constraint in Branch and Main canal
             for(ii=0; ii<CYCLE; ii++){
                    sumSecondaryFlow[ii] = 0.0f;
              for(j = 0; j <NSCHEME; j++){
                            B_GO[ii][j] = Bflowrate[j]*IFLAG[j][ii];
                            sumSecondaryFlow[ii] = sumSecondaryFlow[ii] + B_GO[ii][j];
                     }
                    }
             for(ii=0; ii<CYCLE; ii++){
                            if(sumSecondaryFlow[ii] > (main_capa*PercentFlow/100)){
                                    MnCapaCon[ii] = R2*(sumSecondaryFlow[ii] -
main_capa*PercentFlow/100);
                            }
                     else MnCapaCon[ii] = 0.0f;
                            sumMCapaCon[i] = sumMCapaCon[i] + (float)pow(MnCapaCon[ii],2);
                    }
    ftnss[i] = (float)(sumsupply[i] + sumEquity[i] + sumMCapaCon[i]);
           sumftnss = sumftnss + ftnss[i];
             if(bestfit  > ftnss[i]){
                            bestfit          = ftnss[i];
                            psumLWP          = sumLWP[i];
                            psumOVR          = sumOversupl[i];
                            psumMCapaCon = sumMCapaCon[i];
                            psumsupply       = sumsupply[i];
                            psumEquity       = sumEquity[i];
                            for(j = 0; j<NSCHEME; j++) {
                                    pwara[j] = wara[j];
                                    pshut[j] = shut[j];
                                    pstrt[j] = strt[j];
                                    psumX[j] = sumX[i][j];
                                    psumD[j] = sumD[i][j];
                                    psumsmc_rz[j] = sumsmc_rz[i][j];
                                    psuminfil[j] = suminfil[i][j];
                                    psumEa[j] = sumEa[i][j];
```

```
                                        psumETc[j] = sumETc[i][j];
                                        psumER[j] = sumER[i][j];
                                        psumSMCend[j] = sumSMCend[i][j];
                                        psumO[j] = sumO[i][j];
                                        pstress[j] = cum_stress[i][j];
                                        pGlobalSMC[j] = GlobalSMC[i][j];
                                        psumIrr_day[j] = sumIrr_day[i][j];
                                        pcycle[j] = cycle[i][j];
                                for(ii = 0; ii<CYCLE; ii++) {
                                        pFlag[j][ii] = IFLAG[j][ii];
                                        pETc[j][ii] = ETc[j][ii];
                                        pEa[j][ii] = Ea[j][ii];
                                        psupply[j][ii] = supply[j][ii];
                                        pdemand[j][ii] = demand[j][ii];
                                        psmc_rz[j][ii] = smc_rz[j][ii];
                                        pSMCline[j][ii] = SMCline[j][ii];
                                        pOvr_supl[j][ii] = Ovr_supl[j][ii];
                                        pLWP[j][ii] = LWP[j][ii];
                                        ptsum[j][ii] = tsum[j][ii];
                                        prnWP_RZ[j][ii] = WP_RZ[j][ii];
                                        prnPWP_RZ[j][ii] = PWP_RZ[j][ii];
                                        pFC_RZ[j][ii] = FC_RZ[j][ii];
                                        }//end ii
                                         for(ii=0; ii<CYCLE; ii++) {
                                                pcapaConstr[j][ii] = capaConstr[j][ii];
                                         }
                                 }//end of j forward ploop
                        }// end of if all constraints are satisfied
}//end of i
if (gen == MaxGEN-1 || DELTA_ftnss <= DELTA ) {
index = 999;
   for(j = 0; j<NSCHEME; j++) {
fprintf(f1,"S-%2d flow = %5.3f m3/s \n", j+1, Bflowrate[j]);
                        fprintf(f1,"IFLAG    ");
        for(ii=0; ii<pcycle[j]; ii++)  fprintf(f1,"%9d",pFlag[j][ii]); fprintf(f1,"\n");
                        fprintf(f1,"FC       ");
        for(ii=0; ii<pcycle[j]; ii++)  fprintf(f1,"%9.3f",pFC_RZ[j][ii]); fprintf(f1,"\n");
                        fprintf(f1,"tsum     ");
                        for(ii=0; ii<pcycle[j]; ii++)  fprintf(f1,"%9.3f",ptsum[j][ii]); fprintf(f1,"\n");
        fprintf(f1,"Supply   ");
                        for(ii=0; ii<pcycle[j]; ii++)  fprintf(f1,"%9.3f",psupply[j][ii]); fprintf(f1,"\n");
                        fprintf(f1,"ETc      ");
        for(ii=0; ii<pcycle[j]; ii++)  fprintf(f1,"%9.3f",pETc[j][ii]); fprintf(f1,"\n");
                        fprintf(f1,"Ea       ");
        for(ii=0; ii<pcycle[j]; ii++)  fprintf(f1,"%9.3f",pEa[j][ii]); fprintf(f1,"\n");
                        fprintf(f1,"smc_rz   ");
        for(ii=0; ii<pcycle[j]; ii++)  fprintf(f1,"%9.3f",psmc_rz[j][ii]);fprintf(f1,"\n");
                        fprintf(f1,"SMC      ");
        for(ii=0; ii<pcycle[j]; ii++)  fprintf(f1,"%9.3f",pSMCline[j][ii]); fprintf(f1,"\n");
                        fprintf(f1,"WP       ");
        for(ii=0; ii<pcycle[j]; ii++)  fprintf(f1,"%9.3f",prnWP_RZ[j][ii]); fprintf(f1,"\n");
                        fprintf(f1,"PWP      ");
        for(ii=0; ii<pcycle[j]; ii++)  fprintf(f1,"%9.3f",prnPWP_RZ[j][ii]); fprintf(f1,"\n");
                        fprintf(f1,"BelowWP  ");
        for(ii=0; ii<pcycle[j]; ii++)  fprintf(f1,"%9.3f",pLWP[j][ii]); fprintf(f1,"\n");
                        pEaETcRatio[j] = psumEa[j]/psumETc[j];
                        fprintf(f1,"Scheme  %4d\n",  j+1);
                        fprintf(f1,"Irrifrq %4.1f\n", N_fixloss);
```

```
                    fprintf(f1,"Irr_day %4.1f\n", psumIrr_day[j]);
                    fprintf(f1,"Lag    %4.1f\n", pstrt[j]);
                    fprintf(f1,"Open   %4.1f\n", pwara[j]);
                    fprintf(f1,"shut   %4.1f\n", pshut[j]);
                    fprintf(f1,"Global %9.3f\n", pGlobalSMC[j]);
                    fprintf(f1,"SMCstr %9.3f\n", psumsmc_rz[j]);
                    fprintf(f1,"SMCend %9.3f\n", psumSMCend[j]);
                    fprintf(f1,"Irriga %9.3f\n", psumX[j]);
                    fprintf(f1,"ETc    %9.3f\n", psumETc[j]);
                    fprintf(f1,"Ea_    %9.3f\n", psumEa[j]);
                    fprintf(f1,"Ea/ETc %9.3f\n", pEaETcRatio[j]);
                    fprintf(f1,"ER     %9.3f\n", psumER[j]);
                    fprintf(f1,"infilt %9.3f\n", psuminfil[j]);
                    fprintf(f1,"Drainage%9.3f\n", psumO[j]);
                    fprintf(f1,"stress %9.3f\n", pstress[j]);

            } //end of if j
  }// end of if loop
        avgftnss = sumftnss/POPSIZE;
        minftnss  = ftnss[0];
        for(i=0; i<POPSIZE; i++) {
                if(ftnss[i]  < minftnss)   minftnss  = ftnss[i];
                if(ftnss[i]  < ftnss[POPSIZE]) ftnss[POPSIZE] = ftnss[i];
        }
}
/*********************************/
void Tournament() {
        int ii,jj,j,h, mem;
        for(mem=0; mem < POPSIZE; mem++){
                ii = RandInt(0,POPSIZE-1);
                jj = RandInt(0,POPSIZE-1);
                if(ftnss[ii] <= ftnss[jj]){
                        for(j=0; j<NSCHEME; j++){
                                for(h=0; h<LENGTH; h++){
                                        mated_pop[mem][j][h] = initial_pop[ii][j][h];
                                }
                        }
                }
      if(ftnss[ii] > ftnss[jj]){
                        for(j=0; j<NSCHEME; j++){
                                for(h=0; h<LENGTH; h++){
                                        mated_pop[mem][j][h] = initial_pop[jj][j][h];
                                }
                        }
                }
        }
}// end of tournament
/*********************************/
// Routine to crossover population
void XOVER(int i1, int i2){
        int j, h, kk;
        kk = RandInt(1,LENGTH-1);
  for(j = 0; j<NSCHEME; j++){
          for(h =kk; h<LENGTH; h++)  {
                SwapRealValues(&mated_pop[i1][j][h],&mated_pop[i2][j][h]);
          }
  }
}
```

```
/*********************************/
void UNIXOVER(int i1, int i2){
int j, h, kk;
 for(j= 0; j<NSCHEME; j++){
        for(h = 0; h<LENGTH; h++)  {
                        kk= rand()&01;
                        if(kk==1) SwapRealValues(&mated_pop[i1][j][h],&mated_pop[i2][j][h]);
   }
  }
}
/*************************************/
// Routine to crossover population, 2-site
void TPXOVER(int i1, int i2){
        int j,h,kk1,kk2;
        kk1 = RandInt(1,LENGTH-1);
        kk2 = RandInt(1,LENGTH-1); // this is for 2-site XOVER
        if (kk1>kk2) SwapIntValues(&kk1,&kk2); // this is to swap 2-site if the later is less
 for(j= 0; j<NSCHEME; j++){
        for(h=kk1;h<kk2;h++)  {
                SwapRealValues(&mated_pop[i1][j][h],&mated_pop[i2][j][h]);
   }
  }
}
/*************************************/
void SelectMate(){
   int  j, h, mem, num_select, one;
   float Prob_X;
   num_select = 0;
        for (mem = 0; mem< POPSIZE ; mem++) {
                Prob_X = rand()%1000/1000.0f;
                if (Prob_X < PXOVER){
                        ++num_select;
                        if(num_select%2 == 0){
                                XOVER(one,mem);
                        }
                        else one = mem;
                }
        }
}
/*****************************/
void UniSelectMate(){
        int  j, h, mem, num_select,one;
   float Prob_X;
   num_select = 0;
        for (mem = 0; mem< POPSIZE ; mem++) {
                Prob_X = rand()%1000/1000.0f;
                if (Prob_X < PXOVER){
                        ++num_select;
                        if(num_select%2 == 0){
                                UNIXOVER(one,mem);
                        }
                        else one = mem;
                }
        }
}
/*****************************************/
// this is to radom probability of mutation
void mutatepop(){
```

```
 int i,j,h,mutant;
 double r;
 for (i = 0;i< POPSIZE; i++) {
  for(j = 0; j<NSCHEME; j++){
        r = rand()%1000/1000.0;
        if(r<Pmutate){
        mutant = rand()&01;
        if(mutant == 1) mated_pop[i][j][0] = (float)(mated_pop[i][j][0] + inc);
        else mated_pop[i][j][0] = (float)(mated_pop[i][j][0] - inc);
        }
 for(h= 1; h<LENGTH; h++){
        r = rand()%1000/1000.0;
        if(r<Pmutate){
        mutant = rand()&01;
        if(mutant == 1) mated_pop[i][j][h] = (float)(mated_pop[i][j][h] + inc2);
        else mated_pop[i][j][h] = (float)(mated_pop[i][j][h] - inc2);
        }
        }
        if(mated_pop[i][j][0] < MinLag) mated_pop[i][j][0] = (float)(MinLag);
        else if(mated_pop[i][j][0] > MaxLag-1) mated_pop[i][j][0] = (float)(MaxLag-1);
         for(h= 1; h<LENGTH; h++){
                if(mated_pop[i][j][h] < Min_wrbd) mated_pop[i][j][h] = (float)(Min_wrbd);
                else if(mated_pop[i][j][h] > Max_wrbd) mated_pop[i][j][h] = (float)(Max_wrbd);
         }
        }
         for(j=0;j<NSCHEME;j++)  {
                for(h = 0; h<LENGTH; h++){
                                initial_pop[i][j][h] = mated_pop[i][j][h];
                }
         }
         }//end of i POPSIZE
}
/*************************************/
// Routine to print error message
void nrerror(char *messg){
        puts(messg) ;
  exit(1) ;
}
/*************************************/
// Routine to swop value
void SwapIntValues(int *x, int *y){
        int temp;
        temp = *x;
        *x=*y;
        *y=temp;
}
/*************************************/
// Routine to swop real value
void SwapRealValues(float *xx, float*yy){
        float temp;
        temp = *xx;
        *xx = *yy;
        *yy = temp;
}
/*************************************/
```

APPENDIX H DATA INPUT FOR HETAO IRRIGATION PROJECT

This Appendix presents data for Hetao Irrigation Project which is in Table H-1, and data file input for GA formulation for 12 hour time block is shown in Table H-2.

Table H-1 Data for Xi Li Submain, Hetao Irrigation Project (after Reddy et al. 1999)

Secondary Number	Capacity m^3/s	Operation Time (h)	Volume per canal (m^3)	Time Periods Required	Pre-specified starting time period	Result of Optimal starting time period
(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	0.4	50	72000	4	1 to 5	1
2	0.8	72	207360	6	1 to 5	1
3	0.4	49	70560	4	1 to 5	1
4	0.5	46	82800	4	1 to 5	1
5	0.8	74	213120	6	1 to 5	1
6	0.4	40	57600	3	1 to 5	1
7	1.0	85	306000	7	1 to 5	1
8	0.6	41	88560	3	1 to 5	1
9	0.4	32	46080	3	1 to 5	1
10	0.5	39	70200	3	1 to 5	1
11	0.6	67	144720	6	3 to 7	3
12	1.0	133	478800	11	2 to 6	2
13	0.4	58	83520	5	2 to 6	2
14	0.8	64	184320	5	2 to 6	2
15	0.7	57	143640	5	2 to 6	2
16	0.4	57	82080	5	2 to 6	2
17	0.4	42	60480	4	2 to 6	2
18	0.4	32	46080	3	2 to 6	2
19	0.6	81	174960	7	2 to 6	2
20	0.4	14	20160	1	2 to 6	2
21	0.4	66	95040	6	3 to 7	3
22	1.0	112	403200	9	3 to 7	3
23	1.0	123	442800	10	3 to 7	3
24	0.5	63	113400	5	3 to 7	3
25	0.6	74	159840	6	3 to 7	3
26	0.8	80	230400	7	3 to 7	3
27	0.4	40	57600	3	3 to 7	3
28	0.9	83	268920	7	3 to 7	3
29	1.0	99	356400	8	4 to 8	4
30	1.0	104	374400	9	4 to 8	4
31	0.6	52	112320	4	4 to 8	4
32	0.4	35	50400	3	4 to 8	4
33	0.5	62	111600	5	4 to 8	4

Secondary Number (cont.) (1)	Capacity m^3/s (2)	Operation Time (h) (3)	Volume per canal (m^3) (4)	Time Periods Required (5)	Pre-specified starting time period (6)	Result of Optimal starting time period (7)
34	0.4	17	24480	1	4 to 8	4
35	0.7	96	241920	7	4 to 8	4
36	0.4	51	73440	4	4 to 8	4
37	0.3	15	16200	1	4 to 8	4
38	0.4	23	33120	2	4 to 8	4
39	0.4	33	47520	3	5 to 9	5
40	0.5	41	73800	3	5 to 9	5
41	0.4	42	60480	4	5 to 9	5
42	0.5	39	70200	3	5 to 9	5
43	0.6	58	125280	5	5 to 9	5
44	0.6	58	125280	5	5 to 9	5
45	0.8	75	216000	6	5 to 9	5
46	0.5	48	86400	4	5 to 9	5
47	0.5	77	138600	6	5 to 9	6
48	1.0	118	424800	10	5 to 9	7
49	0.4	31	44640	3	5 to 9	6
50	0.5	55	99000	5	6 to 10	7
51	0.4	29	41760	3	6 to 10	7
52	0.4	42	60480	4	6 to 10	7
53	0.5	62	111600	5	6 to 10	7
54	0.5	48	86400	4	6 to 10	7
55	0.6	76	164160	6	6 to 10	7
56	0.4	26	37440	2	6 to 10	7
57	0.5	62	111600	5	6 to 10	8
58	0.5	60	108000	5	6 to 10	8
59	0.4	33	47520	3	7 to 11	8
60	0.5	56	100800	5	7 to 11	8
61	0.3	10	10800	1	7 to 11	7
62	0.6	76	164160	6	7 to 11	8
63	0.3	3	3240	1	7 to 11	8
64	0.4	46	66240	4	7 to 11	8
65	0.5	70	126000	6	7 to 11	8
66	0.5	48	86400	4	7 to 11	9
67	0.5	46	82800	4	7 to 11	9
68	0.4	35	50400	3	7 to 11	8
69	0.4	52	74880	4	7 to 11	9
70	1.0	95	342000	8	8 to 12	9
71	0.5	66	118800	6	8 to 12	9
72	0.6	60	129600	5	8 to 12	9
73	0.7	84	211680	7	8 to 12	9
74	0.6	72	155520	6	8 to 12	9
75	0.6	72	155520	6	8 to 12	10
76	1.0	98	352800	8	8 to 12	10
77	0.7	80	201600	7	8 to 12	10
78	0.6	74	159840	6	8 to 12	10
79	0.9	95	307800	8	8 to 12	11

Secondary Number (cont.) (1)	Capacity m^3/s (2)	Operation Time (h) (3)	Volume per canal (m^3) (4)	Time Periods Required (5)	Pre-specified starting time period (6)	Result of Optimal starting time period (7)
80	0.6	71	153360	6	12 to 16	12
81	0.5	52	93600	4	14 to 18	14
82	0.4	32	46080	3	15 to 19	15
83	0.5	58	104400	5	13 to 17	13
84	0.6	59	127440	5	13 to 17	13
85	0.5	59	106200	5	13 to 17	13
86	0.4	46	66240	4	14 to 18	14
87	0.7	72	181440	6	12 to 16	12
88	0.7	71	178920	6	12 to 16	12
89	0.5	56	100800	5	13 to 17	13
90	0.8	88	253440	7	12 to 16	12
91	0.6	69	149040	6	13 to 17	13
92	0.6	75	162000	6	13 to 17	13
93	0.4	40	57600	3	16 to 20	16
94	0.7	71	178920	6	13 to 17	13
95	0.7	71	178920	6	13 to 17	13
96	0.6	68	146880	6	13 to 17	13
97	0.5	53	95400	4	15 to 19	15
98	0.6	60	129600	5	14 to 18	14
99	0.6	66	142560	6	13 to 17	13
100	0.4	55	79200	5	14 to 18	14
101	3.0	278	3002400	23	1 to 23	1
102	0.5	55	99000	5	12 to 16	15
103	0.6	60	129600	5	12 to 16	15
104	0.6	66	142560	6	11 to 15	14
105	0.7	66	166320	6	11 to 15	14
106	0.8	71	204480	6	11 to 15	14
107	0.4	52	74880	4	13 to 17	16
108	0.5	53	95400	4	13 to 17	16
109	0.5	52	93600	4	13 to 17	16
110	0.6	59	127440	5	13 to 17	16
111	0.7	66	166320	6	12 to 16	15
112	0.4	35	50400	3	15 to 19	18
113	0.5	63	113400	5	13 to 17	17
114	0.6	63	136080	5	13 to 17	17
115	0.5	56	100800	5	13 to 17	17
116	0.3	6	6480	1	17 to 21	20
117	0.6	64	138240	5	13 to 17	17

Table H-2 Data file input for Hetao Irrigation Project

File name ga48dat.dat, water scheduling for Hetao Irrigation Project (12 hour time block)

Data File and Comments

Number of secondaries

117

Scheme number

1	2	3	4	5	6	7	8	9	10	11	12
	13	14	15	16	17	18	19	20	21	22	23
	24	25	26	27	28	29	30	31	32	33	34
	35	36	37	38	39	40	41	42	43	44	45
	46	47	48	49	50	51	52	53	54	55	56
	57	58	59	60	61	62	63	64	65	66	67
	68	69	70	71	72	73	74	75	76	77	78
	79	80	81	82	83	84	85	86	87	88	89
	90	91	92	93	94	95	96	97	98	99	
	100	101	102	103	104	105	106	107	108	109	
	110	111	112	113	114	115	116	117			

Supply canal full capacity(m^3/sec.)

22.6

Secondary canals full supply (m^3/sec.)

0.4	0.8	0.4	0.5	0.8	0.4	1.0	0.6	0.4	0.5	0.6	1.0
	0.4	0.8	0.7	0.4	0.4	0.4	0.6	0.4	0.4	1.0	1.0
	0.5	0.6	0.8	0.4	0.9	1.0	1.0	0.6	0.4	0.5	0.4
	0.7	0.4	0.3	0.4	0.4	0.5	0.4	0.5	0.6	0.6	0.8
	0.5	0.5	1.0	0.4	0.5	0.4	0.4	0.5	0.5	0.6	0.4
	0.5	0.5	0.4	0.5	0.3	0.6	0.3	0.4	0.5	0.5	0.5
	0.4	0.4	1.0	0.5	0.6	0.7	0.6	0.6	1.0	0.7	0.6
	0.9	0.6	0.5	0.4	0.5	0.6	0.5	0.4	0.7	0.7	0.5
	0.8	0.6	0.6	0.4	0.7	0.7	0.6	0.5	0.6	0.6	0.4
	3.0	0.5	0.6	0.6	0.7	0.8	0.4	0.5	0.5	0.6	0.7
	0.4	0.5	0.6	0.5	0.3	0.6					

Time Periods Required

4	6	4	4	6	3	7	3	3	3	6	11
	5	5	5	5	4	3	7	1	6	9	10
	5	6	7	3	7	8	9	4	3	5	1
	7	4	1	2	3	3	4	3	5	5	6
	4	6	10	3	5	3	4	5	4	6	2
	5	5	3	5	1	6	1	4	6	4	4
	3	4	8	6	5	7	6	6	8	7	6
	8	6	4	3	5	5	5	4	6	6	5
	7	6	6	3	6	6	6	4	5	6	5
	23	5	5	6	6	6	4	4	4	5	6
	3	5	5	5	1	5					

APPENDIX I SOURCE CODE FOR HETAO IRRIGATION PROJECT

This appendix contains the source code for water scheduling problem used to solve the Hetao Irrigation Project with the free specified starting time period. The source code contains 1 file. This file contains function to define by users. At the header of file where the population size, interval of generation to recheck of the improved fitness, minimum improvement of the fitness within the interval generations, number of scheme, number of time step concerned, chromosome length , increment number to mutate gene are able to modified by user. Users are also able to modify probability of crossover, number of mutation per chromosome and maximum generations in the main program.

Miscellaneous routines required by the main program are included in this file as follows:

"read_inputfiles" is a subroutine for reading the input file.

"initial" to initialise initial population.

"objective" contains the objective function and constraints.

"tournament" contains subroutine for tournament selection.

"selectmate" is a subroutine for one choosing one point (subroutine "pxover") or two point crossover (subroutine "tpxover").

"uniselectmate" is a subroutine for uniform crossover.

"mutatepop" is a subroutine used for uniform mutation.

Table I-1 Contents of the files used in the GA model

File name	**Description**
GA48s9.c	Name of project execute file
Proj48.lst	Name of input project file
Ga48dat.dat	Name of input data file
ga48.out	output file for overall detail

```
//GA48s9.c          GA source code for water scheduling problem of Hetao System
//This source code use for free specified starting time block

#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <process.h>
#include <time.h>

#define POPSIZE         8       // input population size
#define INTERVAL        300     // input interval of generation to recheck of improvement fitness
#define DELTA   0.0001  // input minimum improved of fitness after INTERVAL generations
#define PrintMENU       0       // PrintMENU 1 is to print all data, 0 is no print

#define NSCHEME         117     // input number of schemes
#define CYCLE           24      // use 48 for 6hr-TP,use 24 for 12hr-TP, and use 12 for 24hr
#define LENGTH          117     // input chromosome length
#define inc             1       // increment number to mutate

int     RandInt (int, int);
float   RandReal (float,float);
void    Read_inputfiles();
void    SwapIntValues(int *, int *);
void    SwapRealValues(float *, float *);
void    nrerror(char *messg) ;
void    initial();
void    objective();
void    SelectMate();
void    UniSelectMate();
void    mutatepop();
void    Tournament();

int     MaxGEN;
float   TPreq[NSCHEME];
int     gen, i ,index;

float   NofMu, PXOVER;
float   mated[POPSIZE], bestfit, Pmutate, ftnss[POPSIZE+1], DELTA_ftnss, OLD_minftnss;
float   initial_pop[POPSIZE][LENGTH], mated_pop[POPSIZE][LENGTH];

float   Bcapa[NSCHEME], main_capa;

FILE *f1,*f2,*f3 ;

//MAIN PROGRAM
void main (void)
{
  clock_t start, finish;
  float  duration, TotalTime ;

  TotalTime = 0.0f;
  if(!(f1 = fopen("ga48.out","w"))) {
                printf("can't open output file\n");
                exit(1);
        }
```

```
   Read_inputfiles();

         srand(2);

   MaxGEN      =      2500;   // input maximum generation
   PXOVER      =      0.85f;  // input probability of crossover
   NofMu       =      0.15f;  // input number of mutation per chromosome

//  NofMu       =      0.01    //this is for sensitivity analysis for mutation
//do{
//PXOVER      =      0.5f;   //this is for sensitivity analysis for crossover
//do{

   initial();
   index = 0;
         OLD_minftnss   = 100.0f;
         DELTA_ftnss          = 100.0f;
         bestfit              = 1000000.0f;
         ftnss[POPSIZE]  = 1000000000.0f;
   start = clock();

  for(gen = 0; gen <= MaxGEN; gen++) {

         objective(); if(index == 999) break;
         Tournament();
//       SelectMate();
         UniSelectMate();
         mutatepop();

   if (!((gen+1)%INTERVAL)){
     DELTA_ftnss = (float)fabs((double)((OLD_minftnss - ftnss[POPSIZE])/ftnss[POPSIZE]));
     OLD_minftnss = ftnss[POPSIZE];
         }
  } //End of Generation Loop

  finish = clock();
  duration = (float)(finish - start) / CLOCKS_PER_SEC;
  TotalTime = (float)(TotalTime + duration);
  printf(" Time = %5.2f seconds\n", duration);
  fprintf(f1,"Time = %5.2f seconds \n\n", duration);


//  PXOVER = (float) (PXOVER + 0.05f);
//       }while (PXOVER <= 1.01);

//       NofMu = (float)(NofMu + 0.02f);
//       }while (NofMu <= 0.30);


  (void) fclose (f1);
  printf("End of run  ENJOY!!!! ");

} //MAIN PROGRAM END
void Read_inputfiles()
{
   int   ii, jj, num_schemes,scheme_num[NSCHEME];

         char  title1[80],title2[80],title3[80], title4[80];
```

```
        char  schedulefile[20];


        if(!(f2 = fopen("Proj48.lst","r"))) {   //Proj48.lst for 12hr-TP
    nrerror(" Can not open project file") ;//Prj48_6.lst for 6hr-TP
        }
//Prj48_24.lst      for 24hr-TP
  // start reading file here
        fgets(schedulefile, 20, f2) ;

        for(ii = 0; ii < 20; ii++) {
                if(schedulefile[ii] == '\n') schedulefile[ii] = '\0' ;
        }
        (void) fclose(f2) ;

  // now read water scheduling data
        if(!(f2 = fopen(schedulefile,"r"))) {
    nrerror(" Unable to open schedulefile") ;
  }

        // Canal data file start here **********************
  fgets(title1, 80, f2);        // Canal data title
//      puts(title1);

  fscanf(f2,"%d",&num_schemes);  // scan number of schemes here
//      printf("%3d\n",num_schemes) ;  //3

  for(jj=0;jj<num_schemes;jj++) fscanf(f2,"%d ", &scheme_num[jj]);
// for(jj=0;jj<num_schemes;jj++) printf("%6d",scheme_num[jj]);
// printf("\n");


  fgets(title2, 80, f2);        // Main & lateral canals title
// puts(title2);

        fscanf(f2,"%f ", &main_capa) ;  // scan main canal supply
//      printf("%6.2f\n",main_capa) ; // 0.50

        fgets(title3, 80, f2);
// puts(title3);

  // scan branch canal supply
        for(jj = 0; jj < num_schemes; jj++) fscanf(f2,"%f ", &Bcapa[jj]);
        fscanf(f2,"\n");
// for(jj = 0; jj < num_schemes; jj++) printf("%6.2f",Bcapa[jj]);
// printf("\n");
//getchar();

  fgets(title4, 80, f2);        // Main & lateral canals title
// puts(title4);
        for(jj = 0; jj < num_schemes; jj++) fscanf(f2,"%f ", &TPreq[jj]);
        fscanf(f2,"\n");
// for(jj = 0; jj < num_schemes; jj++) printf("%6f",TPreq[jj]);
// printf("\n");
//getchar();

  (void)fclose(f2);
}
```

```
/*********/
float RandReal (float lower,float upper)
{
        float val;
        float nb = ( upper - lower );
        val =  (float)(rand()%1000/1000.0*nb + lower);
        return ((float)val);
}
/*********/
int RandInt (int lower,int upper)
{
        int val;
        int nb = (upper - lower );
        val =  (rand()%nb + lower);
        return val;
}
/*********/
void initial()
{
  int       i, j;
  float nd = RAND_MAX;

   // for random starting time period for each lateral which is decision variable
        // of the problem
  for(i = 0; i<POPSIZE; i++) {
                for(j = 0; j< LENGTH; j++){
                        initial_pop[i][j] = (float)RandInt(0,CYCLE-1);          // random loop

                }
  }
  /***********/
        if( PrintMENU == 1) {
                printf("Gen.  0\n");
                fprintf(f1,"Gen.0\n");
                fprintf(f1,"Initial Population\n");

                for(i=0;i< POPSIZE;i++) {
                        fprintf(f1,"%3d)", i+1);
                        for(j = 0; j< LENGTH ; j++){
                                        fprintf(f1,"%4.0f", initial_pop[i][j]);
                                }
                                fprintf(f1," \n");
                }
                fprintf(f1,"\n");
        }//if printmenu
}
/************************/
void  objective()
{
        int             i, jj, ii,IFLAG[NSCHEME][CYCLE],pFlag[NSCHEME][CYCLE];

        float   sumftnss, minftnss, avgftnss;
        float   branchGO[CYCLE][NSCHEME],
sumBflow[CYCLE],sumCapaConstr[POPSIZE];//
        float   capaConstr[CYCLE],psumBflow[CYCLE], sumobjflow[POPSIZE];//
        float   psumoutflow, outflow[CYCLE], sumoutflow[POPSIZE], poutflow[CYCLE];
        float   pcapaConstr[CYCLE],psumCapa, objflow[CYCLE], pinitial[NSCHEME];
```

```
 sumftnss = 0.0f;
for(i=0;i<POPSIZE;i++) {
 ftnss[i] = 0.0f;
 sumoutflow[i] = 0.0f;
 sumobjflow[i] = 0.0f;
 sumCapaConstr[i] = 0.0f;

 Pmutate  = (float)(NofMu/LENGTH);      //new

 for(jj = 0; jj <NSCHEME; jj++) {            //CYCLE = 24 so CYCLE-1 = 23 count 0-23

                if(initial_pop[i][jj]+TPreq[jj] > CYCLE-1) initial_pop[i][jj] = (float)(CYCLE-
TPreq[jj]);
 }

 for(jj = 0; jj <NSCHEME; jj++){
        for(ii = 0; ii<CYCLE; ii++){
                if(ii < initial_pop[i][jj]) IFLAG[jj][ii] = 0;// ii < randomised TP so ii = 0
                // if ii is in between randomised TP and TPreq so ii = 1
                if(ii >= initial_pop[i][jj] && ii < initial_pop[i][jj]+TPreq[jj]) IFLAG[jj][ii] = 1;
                if(ii >= initial_pop[i][jj]+TPreq[jj]) IFLAG[jj][ii] = 0;

//              fprintf(f1,"%1d", IFLAG[jj][ii]);
        }// end of jj cycle loop
//    fprintf(f1,"\n");
 }//end of ii scheme
// fprintf(f1,"\n");

 // this is canal capacity constraint
 for(ii=0; ii<CYCLE; ii++){          //new
        sumBflow[ii] = 0.0f;
   for(jj = 0; jj <NSCHEME; jj++) {
                branchGO[ii][jj]= Bcapa[jj]*IFLAG[jj][ii];
                sumBflow[ii]    = sumBflow[ii] + branchGO[ii][jj];
        }
 }

 for(ii=0; ii<CYCLE; ii++){          //new
   if(sumBflow[ii] > main_capa) {
                capaConstr[ii] = (float)(sumBflow[ii] - main_capa);
                outflow[ii]   = 0.0f;
//              printf("%7.2f > %7.2f\n", sumBflow[ii], main_capa);
//              getchar();
        }
   else {
                capaConstr[ii] = 0.0f;
                outflow[ii]   = (float)(main_capa - sumBflow[ii]);
        }
        objflow[ii]   = (float)pow(outflow[ii],2);
        sumoutflow[i] = sumoutflow[i] + (float)(outflow[ii]);

        sumobjflow[i] = sumobjflow[i] + (float)(objflow[ii]);

        sumCapaConstr[i] = sumCapaConstr[i] + (float)(capaConstr[ii]);

 }//end of ii in capacity constraint loop

   ftnss[i] = (sumobjflow[i] + sumCapaConstr[i]);
```

```
    sumftnss = sumftnss + ftnss[i];

            if(bestfit > ftnss[i]){
                        bestfit    = ftnss[i];

                        psumCapa   = sumCapaConstr[i];
                        psumoutflow = sumoutflow[i];

        for(ii=0; ii<CYCLE; ii++) {
                                pcapaConstr[ii] =  capaConstr[ii];

                                  poutflow[ii] =  outflow[ii];
                                 psumBflow[ii] =  sumBflow[ii];

                                for(jj = 0; jj<NSCHEME; jj++) {
                                        pinitial[jj] =  initial_pop[i][jj];
            pFlag[jj][ii] =  IFLAG[jj][ii];
                                }
                        }
                }

}//end of i
// printf("end of POPSIZE**************************************\n");

 if (gen == MaxGEN || DELTA_ftnss <= DELTA)      {
            index = 999;

          fprintf(f1,"Canal#\n");
    for(jj = 0; jj<NSCHEME; jj++) {
                        fprintf(f1,"%3d) ",jj+1);
                        for(ii=0; ii<CYCLE; ii++)  fprintf(f1,"%1d",pFlag[jj][ii]);
                        fprintf(f1,"%7.0f\n", pinitial[jj]+1);
          }// end of scheme
          fprintf(f1,"\n");

          fprintf(f1,"T-Period ");
    for(ii=0; ii<CYCLE; ii++) fprintf(f1,"%7d",ii+1); fprintf(f1,"\n");
          fprintf(f1,"sum_flow ");
          for(ii=0; ii<CYCLE; ii++) fprintf(f1,"%7.2f", psumBflow[ii]); fprintf(f1,"\n");
          fprintf(f1,"outflow  ");
          for(ii=0; ii<CYCLE; ii++) fprintf(f1,"%7.2f", poutflow[ii]); fprintf(f1,"\n");

          fprintf(f1,"\n");

    fprintf(f1,"PX %4.2f Mu %4.3f gen %3d", PXOVER, NofMu, gen);
    fprintf(f1," bestfit %7.1f Sumoutflow %5.3f R1 %2d Capa %5.3f\n",
                        bestfit, psumoutflow, R1,psumCapa);

    printf(" PX %4.2f Mu %4.3f gen %3d", PXOVER, NofMu, gen);
    printf(" bestfit %7.1f R1 %2d Capa %5.3f\n",
                        bestfit, R1, psumCapa);

          // use only to find out ftnss in each chromosome
/*        for(i=0; i<POPSIZE; i++){
                fprintf(f1,"%3d %10.3f\n",gen,ftnss[i]);
          } */

 }// end of if loop
```

```
  avgftnss = sumftnss/POPSIZE;
  minftnss  = ftnss[0];
  for(i=0; i<POPSIZE; i++) {
    if(ftnss[i] < minftnss)   minftnss  = ftnss[i];
          if(ftnss[i]  < ftnss[POPSIZE]){
                  ftnss[POPSIZE] = ftnss[i];
          }
  }
  printf("%3d %10.3f \n",gen+1,bestfit);
  fprintf(f1,"%3d %10.3f \n",gen+1,bestfit);

}

/*********************************/
void Tournament()
{
        int ii,jj,j,mem;

        for(mem=0; mem < POPSIZE; mem++){

                ii = RandInt(0,POPSIZE-1);
                jj = RandInt(0,POPSIZE-1);

                if(ftnss[ii] <= ftnss[jj]){
                        for(j=0; j<LENGTH; j++){
                                mated_pop[mem][j] = initial_pop[ii][j];
                        }
                }
    if(ftnss[ii] > ftnss[jj]){
                        for(j=0; j<LENGTH; j++){
                                mated_pop[mem][j] = initial_pop[jj][j];
                        }
                }
        }
/************/
 if ( PrintMENU == 1) {
 fprintf(f1,"After Tournament Selection \n");
  for(mem=0; mem<POPSIZE; mem++) {
     fprintf(f1,"%3d)", mem+1);
        for(j = 0; j<LENGTH; j++){
                        fprintf(f1,"%4.0f",mated_pop[mem][j]);
                }
     fprintf(f1,"\n");
  }
  fprintf(f1,"end\n");
  fprintf(f1,"\n");
 }
}// end of tournament

/*********************************/
// Routine to crossover population
void XOVER(int i1, int i2)
{
        int j, kk;

        kk = RandInt(1,LENGTH-1);
```

```
  for(j = kk; j<LENGTH; j++){
                    SwapRealValues(&mated_pop[i1][j],&mated_pop[i2][j]);
  }
}
/*********************************/
void UNIXOVER(int i1, int i2)
{
 int j, kk;

 for(j= 0; j<NSCHEME; j++){
                    kk = rand()&01;
                    if(kk==1) SwapRealValues(&mated_pop[i1][j],&mated_pop[i2][j]);

 }

}
/*************************************/
// Routine to crossover population, 2-site
void TPXOVER(int i1, int i2)
{
          int j,kk1,kk2;

          kk1 = RandInt(1,LENGTH-1);
          kk2 = RandInt(1,LENGTH-1); // this is for 2-site XOVER
          if (kk1>kk2) SwapIntValues(&kk1,&kk2); // this is to swap 2-site if the later is less

          for(j=kk1;j<kk2;j++)  {
                    SwapRealValues(&mated_pop[i1][j],&mated_pop[i2][j]);
   }

}
/*************************************/
void SelectMate()
{
          int  j, mem, num_select, one;
   float Prob_X;

   num_select = 0;
          for (mem = 0; mem< POPSIZE ; mem++) {
                    Prob_X = rand()%1000/1000.0f;
                    if (Prob_X < PXOVER){
                              ++num_select;
                              if(num_select%2 == 0){
                                        XOVER(one,mem);
                              }
                              else one = mem;
                    }
          }
/******/
 if(PrintMENU == 1) {
          fprintf(f1,"AFTER CROSSOVER\n");
          for(mem = 0; mem<POPSIZE; mem++) {
      fprintf(f1,"%3d)", mem+1);
                    for(j= 0; j< LENGTH; j++){
                              fprintf(f1,"%4.0f",mated_pop[mem][j]);
                    }
              fprintf(f1,"\n");
   }
```

```
    fprintf(f1,"end\n");
  }
}
/*****************************/
void UniSelectMate()
{
        int   j, mem, num_select,one;
   float Prob_X;

   num_select = 0;
        for (mem = 0; mem< POPSIZE ; mem++) {
                Prob_X = rand()%1000/1000.0f;
                if (Prob_X < PXOVER){
                        ++num_select;
                        if(num_select%2 == 0){
                                UNIXOVER(one,mem);
                        }
                        else one = mem;
                }
        }
        if(PrintMENU == 1) {
        fprintf(f1,"AFTER CROSSOVER\n");
         for(mem = 0; mem<POPSIZE; mem++) {
     fprintf(f1,"%3d)", mem+1);
              for(j = 0; j < LENGTH; j++) {
                   fprintf(f1,"%4.0f",mated_pop[mem][j]);
                        }
                 fprintf(f1,"\n");
         }
    fprintf(f1,"end\n");
        }

}
/*****************************************/
// this is to radom probability of mutation
void mutatepop()
{
 int i,j, mutant;
 double r;

          for (i=0;i<POPSIZE;i++) {
                  for(j=0; j<LENGTH; j++){
                                  r = rand()%1000/1000.0;
                                  if(r<Pmutate){
                                          mutant = rand()&01;
                                          if(mutant == 1) mated_pop[i][j] = (float)(mated_pop[i][j]
+ inc);
                                          else mated_pop[i][j] = (float)(mated_pop[i][j] - inc);


                                          if(mated_pop[i][j] < 0.0f) mated_pop[i][j] = 0.0f;
                                          else if(mated_pop[i][j] > CYCLE) mated_pop[i][j] =
(float)(CYCLE);
                                  }
                  }       // end j

                  for(j = 0; j< LENGTH; j++){
                                  initial_pop[i][j] = mated_pop[i][j];
```

```
                    }
            }//end of i POPSIZE
/*****************/
 if ( PrintMENU == 1) {
 fprintf(f1,"Mutated Population \n");
  for(i=0;i<POPSIZE;i++) {
       fprintf(f1,"%3d)", i+1);
                    for(j = 0; j<LENGTH; j++){
                      fprintf(f1,"%4.0f",initial_pop[i][j]);
                    }
              fprintf(f1,"\n");
  }
  fprintf(f1,"end\n");
  fprintf(f1,"\n");
 }
}
/*************************************/
// Routine to print error message
void nrerror(char *messg)
{
          puts(messg) ;
   exit(1) ;
}
/*************************************/
// Routine to swop value
void SwapIntValues(int *x, int *y)
{
          int temp;
          temp = *x;
          *x=*y;
          *y=temp;
}
/*************************************/
// Routine to swop real value
void SwapRealValues(float *xx, float*yy)
{
          float temp;

          temp = *xx;
          *xx = *yy;
          *yy = temp;
}
/*************************************/
```

Publications

The following papers have been published or submitted for publication in journal or conference.

Submitted in Journal

Wardlaw, R., and Bhaktikul, K. (2000). " Application of a genetic algorithms for water allocation in an irrigation system. *J. Irrigation and Drainage*. ICID, CIID.

Presented in Conferences

Bhaktikul, K. (1999). "Application of a genetic algorithm to water allocation in an irrigation canal system." British Hydrological Society, Ninth Postgraduate Symposium, The Water and Environmental Management Research Centre, Department of Civil and Environmental Engineering, University of Bristol, 29th – 30th March 1999.

Bhaktikul, K. (2000). "Application of a genetic algorithm to water allocation in an irrigation canal system" Scottish Hydraulics Study Group & Scottish Hydrological Group, Postgraduate Research in Hydraulics & Hydrology in Scotland, Strathclyde University, 12th January 2000.

IRRIGATION AND DRAINAGE
Irrig. and Drain. **00**: 0–00 (2001)

APPLICATION OF A GENETIC ALGORITHM FOR WATER ALLOCATION IN AN IRRIGATION SYSTEM[1]

ROBIN WARDLAW* AND KAMPANAD BHAKTIKUL
School of Civil and Environmental Engineering, University of Edinburgh, Scotland, UK

ABSTRACT

Irrigation is the largest use of fresh water globally, and improved management of water within irrigation systems is essential if scarce resources are to be used to their maximum benefit and used in an equitable manner. The management of water allocation in large irrigation systems is important, as even a small improvement in operation may lead to significant benefits.

An optimisation approach based on genetic algorithms (GAs) is described for real time allocation of irrigation water supplies. Appropriate objective functions for the water allocation problem have been derived previously and solved using quadratic programming (QP). This paper describes work on the development of a GA for the water allocation problem. The GA approach is very flexible, and is easily set up for a wide range of linear and non-linear objective functions.

This paper presents the development of a GA for the irrigation water allocation problem and investigates the sensitivity of the approach to different formulations. The GA approach can provide solutions that are similar to those produced by QP. It is, however, sensitive to string length, and has difficulty in meeting closed nodal water balance constraints. It is concluded that the GA approach offers no advantage over the QP for the water allocation problem. Copyright © 2001 John Wiley & Sons, Ltd.

RÉSUMÉ

L'irrigation est, dans le monde, la manière la plus répandue d'exploiter l'eau douce et l'amélioration de la gestion de l'eau, au sein des systèmes d'irrigation, est essentielle si l'on veut utiliser de façon équitable les ressources peu abondantes au maximum de leur bienfait. La gestion de la répartition de l'eau dans les grands systèmes d'irrigation est donc importante: la mise en place d'une infime amélioration peut en effet engendrer des bénéfices considérables.

Une approche d'optimisation, basée sur les algorithmes génétiques (AGs) pour une répartition en temps réel des provisions d'eau pour l'irrigation, est exposée. D'adéquates fonctions objectives pour le problème de la répartition de l'eau ont auparavant été identifiées et résolues en utilisant la programmation quadratique (PQ). Notre article décrit le travail réalisé lors du développement d'un AG concernant le problème de répartition de l'eau. L'approche proposée par l'AG est très souple et peut facilement être mise en place pour une grande variété de fonctions objectives linéaires ou non.

* Correspondence to: Division of Engineering, School of Civil and Environmental Engineering, The University of Edinburgh, Crew Building, The King's Buildings, Edinburgh EH9 3JN, Scotland, UK.
[1] Application d'un algorithme génétique pour la répartition de l'eau dans un système d'irrigation.

Accepted ~~18 August 1999~~ 27 November 2000

Notre article présente le développement d'un AG pour le problème de répartition de l'eau destinée à l'irrigation et examine la sensibilité de cette approche à des formulations différentes. L'approche AG peut fournir en fait des solutions semblables à celles issues de la PQ. Mais elle est toutefois sensible à la longueur des séries et éprouve des difficultés à répondre aux restrictions imposées par l'étroit équilibre nodal de l'eau. En conclusion, il apparaît que l'approche AG n'offre pas plus d'avantages que la PQ en ce qui concerne le problème de la répartition de l'eau.

INTRODUCTION

Recent studies by Wardlaw and Barnes (1996, 1997, 1998) have been concerned with the real time allocation of water supplies in irrigation systems with complex distribution networks. Wardlaw and Barnes developed a quadratic programming (QP) approach to optimise water allocation. Genetic algorithms (GAs) were identified as a possible alternative approach, and one that might be more robust than QP. GAs were first introduced by Holland (1971), and in 1975 he published the book entitled *Adaptation in Natural and Artificial Systems* (Holland, 1975) which for many years was a standard reference. Since then, GAs have been applied to a wide range of problems.

A review of GA work related to irrigation water distribution systems identified only one previous application (Chen, 1997). The objective of that investigation was to optimise the schedule of reservoir releases to maximise economic benefits. The study found that the conventional GA approach was not appropriate with the large number of variables used. Modifications to the conventional GA approach were required to improve the rate of convergence.

BASIS OF GENETIC ALGORITHMS

A GA is a random search algorithm. It is a robust method for searching the optimum solution to complex problems, even though it may not always necessarily lead to the best possible solution (Goldberg, 1989). In a GA, the problem is represented by a population of strings (or chromosomes in the biological terminology). Each string comprises a number of blocks, which represent the individual variables of the problem (genes in biological terminology). The variables represented in the string can be processed in an evaluation function, or fitness function which is in effect the objective function.

Decision variables can be represented in a GA in a number of ways. In conventional GAs binary coding is normally used. In binary coding, each block, or gene, is further broken down into a series of binary digits. Gray coding is a variant of binary coding that can offer improved solutions. Real-value representation in which genes represent a single variable as a real number has been shown to offer advantage over both binary and gray coding (Wardlaw and Sharif, 1999).

Strings are processed and combined according to their fitness in order to generate new strings combining the best features of two parent strings. Strings with the highest fitness have the greatest chance of contributing to future generations, as in the process of natural selection. Excellent introductions to GAs are given by Goldberg (1989) and by Michalewicz (1992).

Fundamental operators of genetic algorithms

There are three basic operations involved in manipulating strings and moving to a new generation. These are selection, crossover and mutation. The approach taken to the operators of selection, crossover and mutation can influence the results obtained, and different problems may require different approaches.

(a) Selection. The selection operator is that through which strings are selected for inclusion in the reproduction process and for participation in the next generation. The fittest strings have the highest probability of being used in reproduction. There are a number of approaches to selection, all of which determine the probability of selection as a function of fitness. A brief review of alternative selection schemes has been given by Wardlaw and Sharif (1999), from which it is clear that tournament selection (Goldberg and Deb, 1991; Yang and Soh, 1997) offers a means of maintaining diversity in the population. In tournament selection, rather than performing selection on the basis of fitness within the whole population, groups of individuals are selected at random, and the individuals within these groups having the highest fitness are selected for inclusion in the next generation. A number of rank selection schemes are in use (Michalewicz, 1992) and these tend to ensure that good chromosomes have better chances to be selected for the next generation. Modifications are available to ensure that the fittest chromosome will always be duplicated to the next generation. This is called the elitist strategy, and ensures that the best member of the population will always be used in reproduction (Davis, 1991).

(b) Crossover operator. Crossover is an extremely important part of a GA. If crossover is neglected, the result is no longer a genetic algorithm and the performance will be degraded (Davis, 1991). The crossover operator permits the exchange of genes between pairs of chromosomes in a population. This operator offers the possibility of good genetic material from different individual strings being combined to create an even fitter individual. The operator is intended to preserve the best material from two parent strings. The number of strings in which material is exchanged is controlled by the crossover probability, which is an input to a GA. The crossover probability is normally in the range of 0.5–1.0. Three approaches to crossover are described by Goldberg (1989) and Michalewicz (1992): one-point crossover, two-point crossover and uniform crossover. An excellent summary of these crossover operators has been given by Wardlaw and Sharif (1999).

(c) Mutation operator. Mutation is an important operator of GAs that permits new genetic material to be introduced to a population. A mutation probability can be specified that permits random mutation. In binary representation, individual alleles or bits of a gene simply have their values changed from a 0 to 1 or vice versa. For real-value representation Michalewicz (1992) has outlined two basic approaches to mutation. These are uniform mutation and non-uniform mutation. In uniform mutation, the value of a gene can be mutated randomly within its feasible range of values, while modified uniform mutation permits modifications of a gene by a specified amount. In non-uniform mutation, the amount by which genes can be mutated is reduced as the run progresses, thereby reducing the risk of disturbing good solutions.

THE WATER ALLOCATION PROBLEM

The real time water allocation problem addressed is one of ensuring the equitable distribution of irrigation water supplies within an irrigation system. It is neither a planning problem, in that

crops are assumed to be in the ground, nor is it a scheduling problem, in that irrigation supplies are assumed to be run of river and in scheme scheduling is not considered.

The objective function

Wardlaw and Barnes (1998) have demonstrated that the objective function that preserves equity may be written as

$$\text{Minimise } Z = \sum_{i=1}^{n} \frac{(d_i - x_i)^2}{d_i} \tag{1}$$

where n is the number of irrigation schemes, d_i the irrigation demand for scheme i and x_i the irrigation supply to scheme i.

Three basic constraints must be satisfied in finalising a solution to the problem. These are the reach capacity constraint, the nodal water balance constraint and the scheme supply constraint. Referring to Figure 1, the constraints may be defined as follows:

1. *Capacity constraint:*

$$Q_{ij} \leq qmax_{ij} \tag{2}$$

where Q_{ij} is the flow to node j from node i and $qmax_{ij}$ the maximum capacity of canal connecting nodes i and j (or reach capacity).

2. *Nodal balance constraint:*

$$Qinf_i + \sum_{j=1}^{m} Q_{ij} - x_i - Qsink_i = 0 \tag{3}$$

where i, j are the node references, $Qinf_i$ the external inflow to node i (i.e. not from the canal system), Q_{ij} the flow in canal connecting nodes, x_i the water supply to irrigation scheme i, $Qsink_i$ the outflow to system sink and m the number of reaches connected to node i.

3. *Supply constraint:*

$$x_i \leq d_i \tag{4}$$

where d_i is the irrigation demand of scheme i.

Penalty functions and penalty factors

In a GA, it is necessary to introduce penalty functions to ensure that the system constraints are satisfied. In effect, the requirement is to demote or exclude chromosomes that do not satisfy the system constraints. There are three components to the penalty function used for the water allocation problem as shown below:

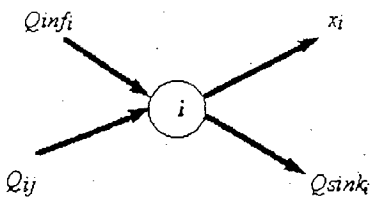

Figure 1. Nodal water balance components

1. *If* $Q_{ij} > qmax_{ij}$

$$P_1 = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{|Q_{ij} - qmax_{ij}|}{qmax_{ij}} \tag{5}$$

where n is the number of nodes in the system.

2. *If nodal balance* > 0.001

$$P_2 = \sum_{i=1}^{n} \frac{\left| Qinf_i + \sum_{j=1}^{m} Q_{ij} - x_i - Qsink_i \right|}{Qinf_i + \sum_{j=1}^{r} Q_{ij(in)}} \tag{6}$$

where $Q_{ij(in)}$ is the flow(s) into node i and r the number of reaches flow into node i.

3. *If* $x_i > d_i$

$$P_3 = \sum_{i=1}^{n} \frac{|x_i - d_i|}{d_i} \tag{7}$$

where n is the number of irrigation schemes.

These penalty functions could be used in quadratic form in order to increase the sensitivity to violation (Goldberg, 1989). A penalty factor (R) is used to weight the penalty functions and the objective function may be rewritten as

$$\text{Minimise } Z = \sum_{i=1}^{n} \frac{(d_i - x_i)^2}{d_i} + R_1 \cdot P_1 + R_2 \cdot P_2 + R_3 \cdot P_3 \tag{8}$$

DEVELOPMENT OF A GA FOR WATER ALLOCATION

There are two fundamental approaches to setting up a GA for the water allocation problem. The first is to represent only canal flows in the chromosome, and from there to derive irrigation supply from the nodal water balance. The second approach is to include both canal flows and irrigation supplies in the chromosome, making irrigation supply a decision variable also. Both approaches have been tested in application to a simple irrigation network. Analysis has also been carried out to determine the most appropriate representation scheme.

TESTING ON A SIMPLE NETWORK

Preliminary development of the GA approach was carried out on the simple network shown in Figure 2. The maximum capacity in each reach was set at 15 $m^3 s^{-1}$. Initially the inflow (*Qinf*) was set to 15 $m^3 s^{-1}$, and the demands *d1*, *d2*, *d3*, *d4*, *d5* and *d6* were set to 0, 4, 5, 5, 0 and 3 $m^3 s^{-1}$ respectively. This resulted in water stress and the GA was required to distribute available resources equitably.

Representation schemes

To compare binary and real-value coding, the GA was set up with canal flows as the decision variables represented in the chromosome.

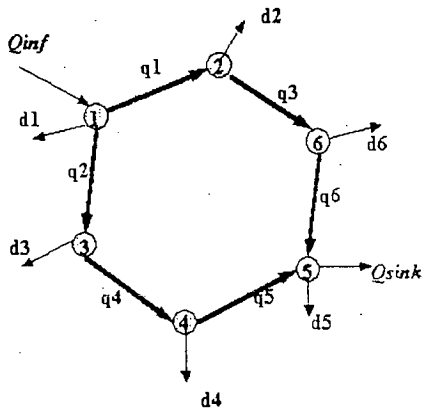

Figure 2. Preliminary test network

(a) Binary coding set-up. An example of a chromosome representing a set of canal flows is shown in Figure 3. Each string has six genes (for six canal flows) and each gene comprises a number of alleles or bits. These alleles are encoded/decoded by linear mapping to their decimal equivalent canal flow. The mutation operator can cause large changes to gene values in binary representation. Gray coding has been found (Dandy *et al.*, 1996) to create fewer disturbances to good solutions, but it has been demonstrated by Wardlaw and Sharif (1999) that real-value coding offers many advantages.

(b) Real-value coding set-up. To initialise the population in real-value coding, flows for each canal in each chromosome in the population are generated randomly within the range of the canal capacities. The variable space is continuous.

Evaluation of results

Both representation schemes achieved near equity in supply. Results are presented as supply/demand ratios in Figure 4. An important difference between the binary and real-value codings is in the length of chromosomes. In binary coding there are five alleles per gene giving a chromosome length of 30. In real-value coding, there is only one allele per gene and the chromosome length is thus 6. A further disadvantage of binary coding is that coding and decoding slow down execution time. The average execution time for 500 generations with binary coding was 5 s while that with real-value coding was 2 s. Figure 5 shows the progress of the GAs towards solution with binary and real-value coding. Real-value coding clearly approaches the solution more quickly. A constraint on binary coding is the accuracy with which flows can be mapped to the binary bits, or alleles in a gene. The linear mapping of values results in an accuracy of 0.00–0.05 $m^3 s^{-1}$ in each canal. To improve accuracy with binary coding, it would be necessary to include more alleles in each gene. This in turn increases string length and can result in slower execution and further problems in maintaining good solutions. The results achieved on this preliminary test demonstrate that GAs are capable of producing acceptable results, and that real-value coding is superior to binary coding.

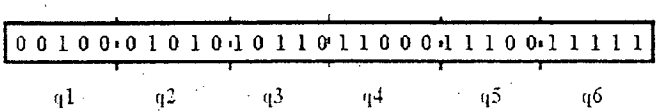

Figure 3. A chromosome representing a set of canal flows (for Figure 2)

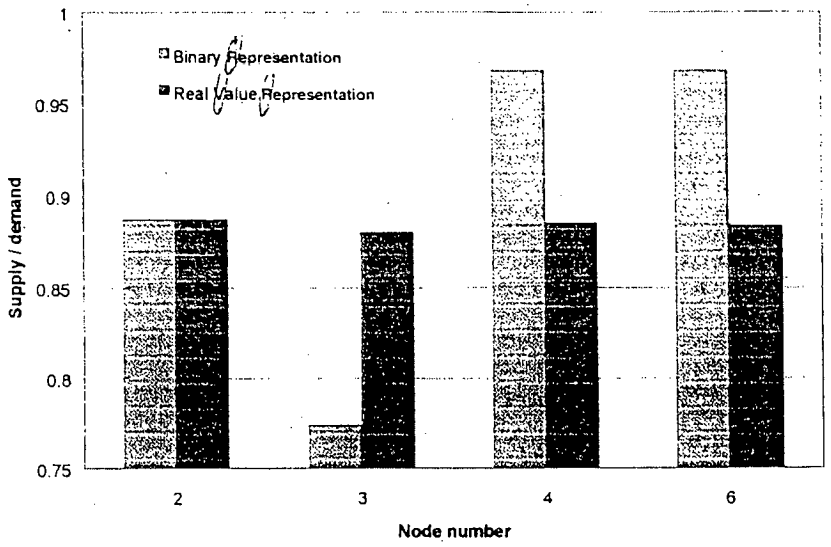

Figure 4. Supply/demand ratios for different representation schemes

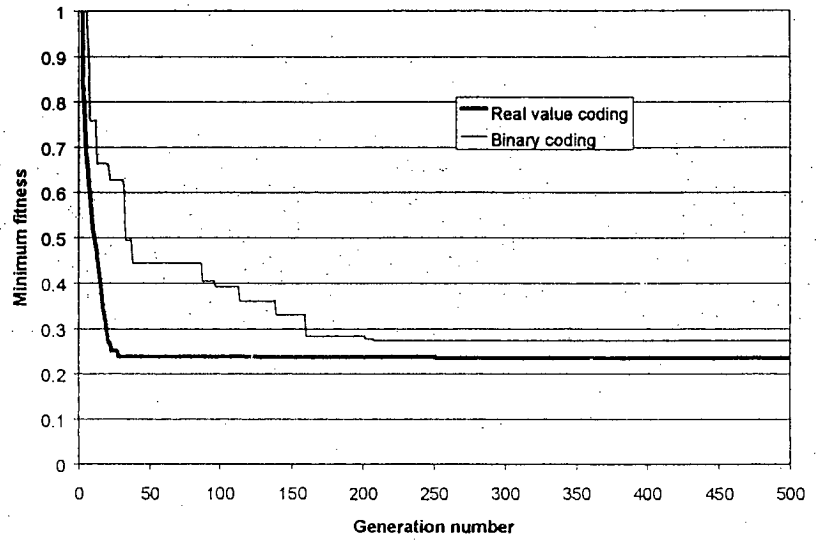

Figure 5. Solution progress with different representation schemes

GA performance with strings comprising irrigation supplies and flows

A GA set up with irrigation supplies and flows represented in a chromosome requires a longer string length, but has the same constraints as a GA set up with flows alone. For the simple network shown in Figure 2, the string comprises 12 genes as there are 6 irrigation supplies. A GA was set up using real-value coding, and results compared with those obtained earlier with a representation consisting of canal flows only. Results are presented in Figure 6. Clearly both approaches produce acceptable results. The GA set up with flows and supplies takes longer to reach an acceptable solution than the GA operating on flows alone, however, as can be seen from Figure 7. The nodal balance constraint for the GA with both the flows and the supplies in a chromosome required an increased penalty factor to ensure that constraints were satisfied.

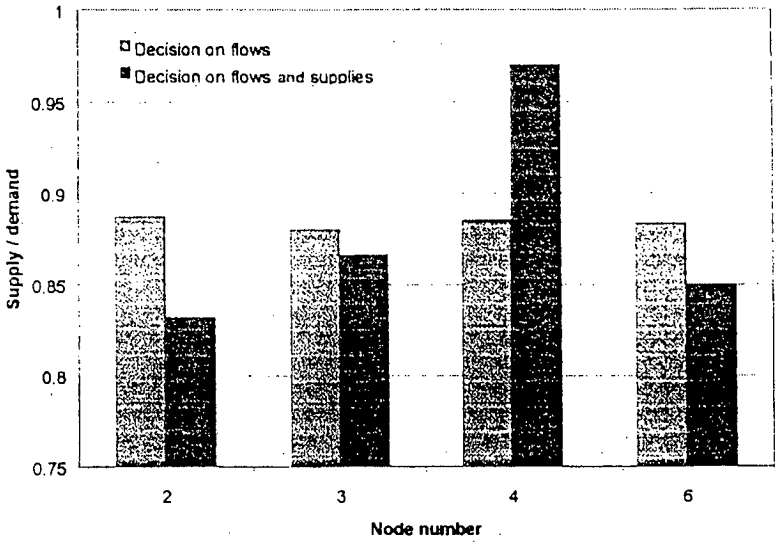

Figure 6. Supply/demand ratios with alternative formulations

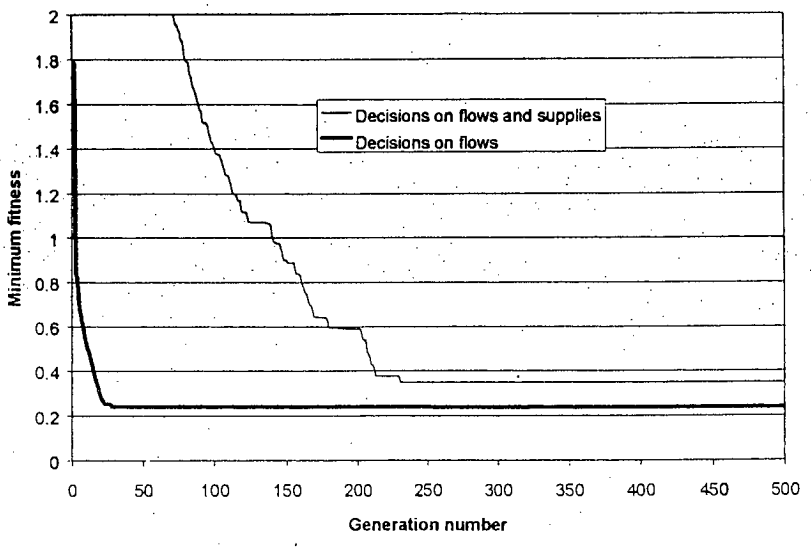

Figure 7. Solution progress with alternative formulations

The reason for this is that random modification to two variables in the same constraint results in a greater chance of a good solution being disturbed. It is desirable to keep string length as short as possible.

SENSITIVITY ANALYSIS ON THE TEST NETWORK

The preceding sections have demonstrated that real-value coding is superior to binary coding, and that the problem is best formulated using canal flows only, keeping irrigation supplies as a dependent variable. On the basis of this preferred set-up, the sensitivity of the GA to crossover probability, mutation probability and population size was investigated. The real-value coding is apparently insensitive to mutation probability, although best results are obtained with about

0.75 mutations per chromosome. This number of mutations per chromosome was used to test the sensitivity of performance to the probability of crossover. The best results are achieved with a crossover probability of 0.95, although acceptable results are obtained over a fairly wide range. From the literature reviewed it has been suggested that the appropriate crossover probability is in the range of 0.5–1 (Goldberg, 1989). It was found that acceptable results could be achieved with all crossover schemes, but that non-uniform mutation performed better than modified uniform mutation.

Acceptable results were obtained within a wide range of population sizes. The best results were achieved for the test network with a minimum population size of 100 with crossover probability of 0.95 and the number of mutations per chromosome of 0.75. With too small a population there will be insufficient diversity, but with too large a population an increased number of generations will be required for the best individuals to emerge.

PRACTICAL APPLICATION OF THE GA

The GA for the water allocation problem has been applied to the irrigation system of Tukad Ayung in Bali, Indonesia. The system has been described by Wardlaw and Wells (1996), and was used by Wardlaw and Barnes (1996) in the development of their QP approach. The network diagram for the system is shown in Figure 8. It comprises 69 reaches and 56 nodes.

Objective function used

The objective function used was basically that of Equation (8). It was found that with this more complex network, the global water balance could accumulate errors from individual nodal balances. A further global water balance constraint was therefore introduced in penalty form:

$$P_4 = \frac{\left(\sum_{i=1}^{n} Qinf_i - \sum_{i=1}^{n} X_i - \sum_{i=1}^{n} Qsink_i \right)}{\sum_{i=1}^{n} Qinf_i} \tag{9}$$

where P_4 is the penalty for the global water balance constraint.

In the Ayung network with 56 nodes at which water balance constraints must be satisfied, it was found that penalty function representation was very important. Appropriate penalty factors were determined by experiment. From the series of model runs it was also found that a quadratic form of the nodal water balance constraint (Equation (6)) best suited the problem.

The problem has been set up with canal flows as the decision variables. Real-value representation, the elitist strategy, uniform crossover and non-uniform mutation were adopted. The GA was set up with a population of 100, probability of crossover of 0.95 and 0.75 mutations per chromosome. It was found that a low error tolerance was required in decision variables as a result of the complexity of the Tukad Ayung system, and the string length involved. In time steps with no water stress, the water balance constraints were satisfied to a precision of 0.008 $m^3 s^{-1}$. In time steps with water stress, the water balance constraints were rarely satisfied to this precision. It was found that nodal balance error was typically in the range of 0.000–0.1.

Average irrigation deficits are compared with those achieved by Wardlaw and Barnes (1997) using QP in Figure 9. The GA1 results refer to this original set-up. QP clearly performs much better than GA1. The average annual irrigation demand per time step is 11.5 $m^3 s^{-1}$. Using QP

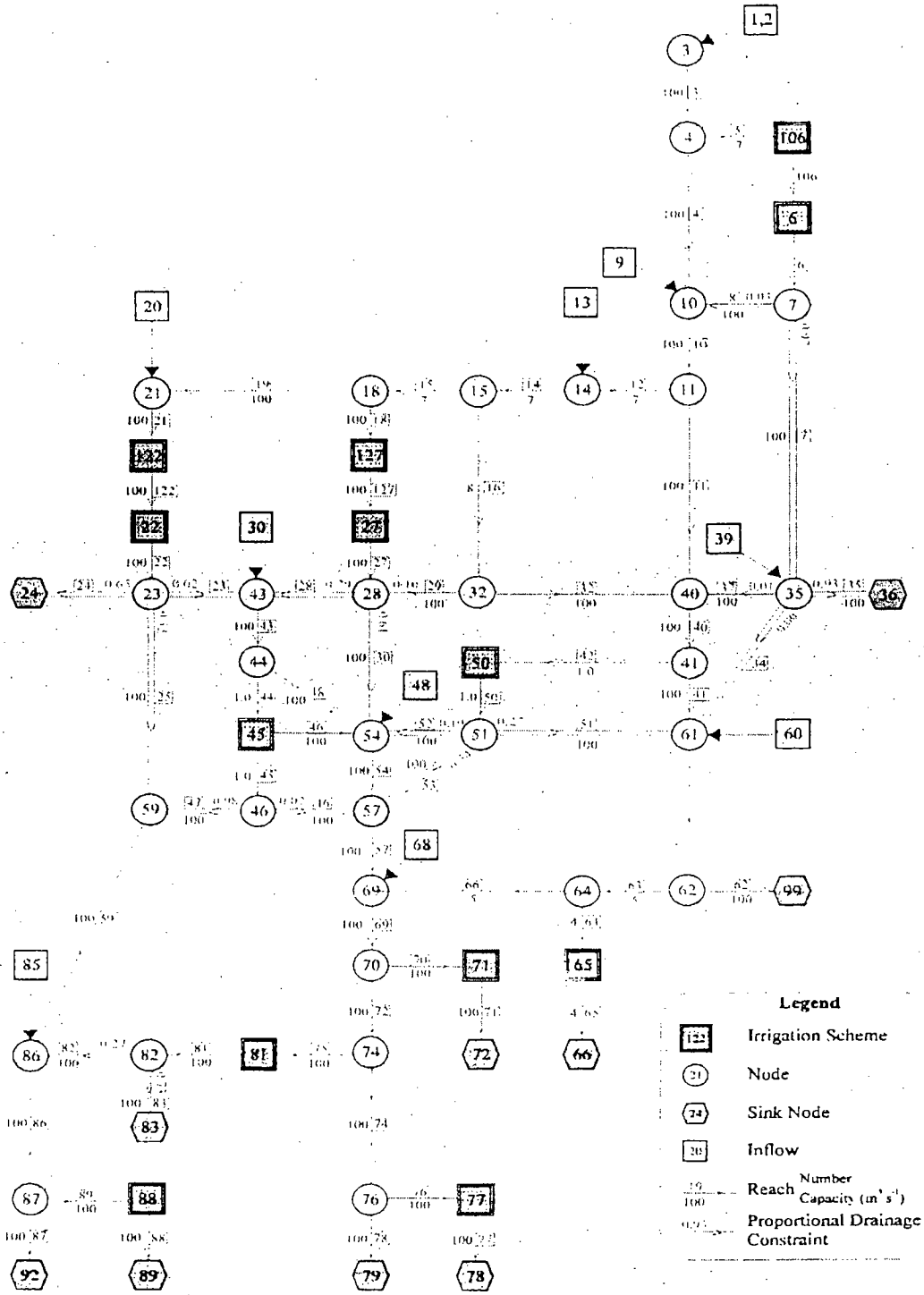

Figure 8. Schematic of the Ayung irrigation system (after Wardlaw and Barnes, 1996)

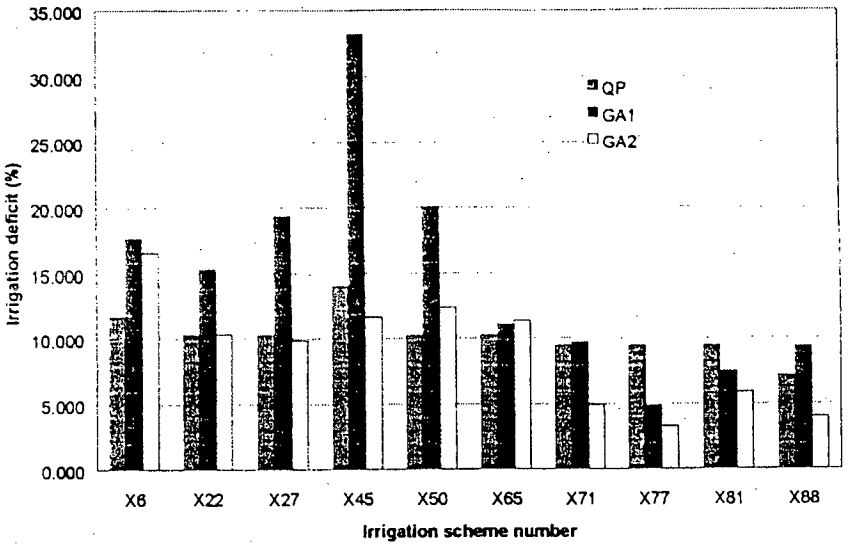

Figure 9. Comparison of GA performance with that of QP

the average supply per time step was 10.27 $m^3 s^{-1}$ while with GA1 the average supply was 9.68 $m^3 s^{-1}$. The GA1 set-up did not perform well.

IMPROVED GA APPROACH

In the original GA formulation (GA1), flows were computed for every reach. However, at many junction nodes in the system, where flows are divided or combined, there is no irrigation demand. At such nodes, flows in one reach can be computed from the nodal water balance, thus reducing the number of decision variables and gene length. In this revised formulation (GA2), the gene length was reduced from 51 to 25 and the potential for constraint violation also reduced considerably. The objective function used was still that of Equation (8), with minor adjustment to constraints and addition of constraint 4 (Equation (9)). With the revised approach, the average supply per time step was increased to 10.12 $m^3 s^{-1}$, which is much closer to the solution achieved by QP. The results are included in Figure 9. Clearly GA2 is a significant improvement on GA1, and near equity has been achieved.

CONCLUSIONS

It has been demonstrated that with the improved GA formulation, solutions to the water allocation problem can be achieved that are similar to those achieved by QP. The GA should be set up with canal flows as the decision variables. The number of decision variables can be reduced by making use of continuity at junctions with no irrigation demands to permit one canal flow to become a dependent variable.

It is concluded that a real-value representation, tournament selection together with the elitist strategy, uniform crossover and non-uniform mutation will produce the best results. A crossover probability of 0.95 and 0.75 mutations per chromosome are suitable for the water allocation problem. The approach has been shown to be robust and not particularly sensitive to crossover or mutation probabilities. It is sensitive to population size, however, and too small a population restricts diversity.

The GA is clearly sensitive to string length, and this will limit its application to larger problems. As string length increases, maintaining good solutions becomes more difficult. The closed nature of the water balance at supply nodes makes solution by GA difficult. The main advantage of the GA approach is that it can be set up with any form of objective function. However, for the water allocation problem, although the improved GA produced acceptable results, it offers no real advantage over the QP approach. Execution times for the QP approach were significantly shorter than those of the improved GA. It is thought that the GA does hold promise for application to water scheduling problems.

REFERENCES

Chen Y-M. 1997. Management of water resources using improved genetic algorithms. *J. Comp. and Electro. in Agri.* **18**: 117–127.

Dandy CG, Simpson AR, Murphy LJ. 1996. An improved genetic algorithm for pipe network optimization. *Water Resour. Res.* 32(2): 449–458.

Davis L. 1991. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold: New York.

De Jong KA. 1975. An analysis of the behaviour of a class of genetic adaptive systems. Doctoral dissertation, University of Michigan, Ann Arbor, Mich.

Goldberg DE. 1989. *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley: Reading, Mass.

Goldberg DE, Deb K. 1991. A comparative analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms*, Morgan Kaufman: San Mateo, Calif.; 69–93.

Holland JH. 1971. Proc. and processors for schemata. In *Associative Information Processing*, Jacks EL (ed.), Elsevier: New York; 127–146.

Holland JH. 1975. *Adaptation in Natural and Artificial Systems*. MIT Press: Cambridge, Mass.

Michalewicz. 1992. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer: New York.

Wardlaw RB, Barnes JM. 1996. *Improved Irrigation System Planning and Management, Optimal Allocation of Irrigation Water Supplies. ODA TDR Research Project No. 6261, Phase I report*, April 1996.

Wardlaw RB, Barnes JM. 1997. *Improved Irrigation System Planning and Management, Optimal Allocation of Irrigation Water Supplies. ODA TDR Research Project No. 6261, Phase II report*, June 1997.

Wardlaw RB, Barnes JM. 1998. *Improved Irrigation System Planning and Management, Optimal Allocation of Irrigation Water Supplies. ODA TDR Research Project No. 6261, Phase III report*, June 1998.

Wardlaw RB, Sharif M. 1999. Evaluation of genetic algorithms for optimal reservoir system operation. *J. Water Resour. Plng and Mgmt*, ASCE 125(1): 25–33.

Wardlaw RB, Wells RJ. 1996. Simulating crop production and resource development impacts: the Lower Ayung simulation model. *Proc. Instn Civ. Engrs Wat., Marit. & Energy* 118(2): June.

Yang JP, Soh CK. 1997. Structural optimization by genetic algorithms with tournament selection. *J. of Comp. in Civ. Engrg*, ASCE 11(3), 195–200.

Unlinked Bibliographic References

AUTHOR PLEASE CITE REFERENCE De Jong (1975) IN TEXT.

Scottish Hydraulics Study Group & Scottish Hydrological Group

Postgraduate Research in Hydraulics & Hydrology in Scotland

12th Jan 2000, Evening Session, 6.30pm

Lecture Room 1, McCance Building, Strathclyde University

Details of the three papers to be presented are given below and in the abstracts on the following pages.

Detailed LDA Studies Of Erosion Control Geomats

D.J. McKAY
Environmental Civil Engineering Division, Department of Energy and Environmental Technology, Glasgow Caledonian University, Glasgow, Scotland, UK

Risk Assessment For Extreme Floods In The UK : A Heterogeneity Approach

A. CARGILL
Environmental Systems Research Group, Department of Geography, University of Dundee, Dundee, Scotland, UK

Application Of A Genetic Algorithm To Water Allocation In An Irrigation Canal System

K. BHAKTIKUL
School of Civil & Environmental Engineering, University of Edinburgh King's Buildings, Edinburgh, Scotland, UK

Application Of A Genetic Algorithm To Water Allocation In An Irrigation Canal System

KAMPANAD BHAKTIKUL
School of Civil & Environmental Eng.
University of Edinburgh
Crew Building
King's Buildings
West Mains Rd.
Edinburgh EH9 3JN
Scotland

Supervisor
R. WARDLAW

Abstract

The demand for water is increasing rapidly in the developing world. The gap between water demand and water supply is widening, making effective water resources management vital. Irrigation represents the largest use of fresh water globally, and improved management of water within irrigation systems is essential if scarce resources are to be used in an equitable manner and to their maximum benefit. The management of water allocation in large irrigation systems is important, as even a small improvement in operation may lead to significant benefits.

An optimisation approach based on Genetic Algorithms (GAs) is described for real time allocation of irrigation water supplies. Appropriate objective functions for the water allocation problem have been derived previously and solved using quadratic programming. There had been concern that the quadratic programming approach may become computationally bounded for large systems. It was also thought that the approach would be difficult to apply to water scheduling problems. This paper describes work on the development of a GA for the water allocation problem. Although GAs have been actively researched for 30 years, only one previous application to an irrigation problem has been found in the literature. The GA approach is very flexible, and is easily set up for a wide range of linear and non-linear objective functions.

This paper presents the development of a GA for the irrigation water allocation problem and investigates the sensitivity of the approach to different formulations. The advantages and the shortcomings of the technique are discussed.